An Empirical Study of the KAS

Analyzing quality requirements and their definitions

TDT4735 Software Engineering, Depth Study

Yngve Halmø
halmo@stud.ntnu.no

Geir-Arne Jenssen
geirarj@stud.ntnu.no

December 19, 2005
Supervisor: Tor Stålhane

NORWEGIAN UNIVERSITY OF TECHNOLOGY AND SCIENCE,
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

Abstract

The main focus of this project is analyzing quality attributes and their definitions in an already developed software system, that has been developed by the authors but has not been taken into use yet. The system is meant to be used by UNN in Tromsø, and if taken into use it will save the user for both time and money. The reason for analyzing this system before it is made use of, is to make sure the required quality requirements is fulfilled, and that the system will fit the user's needs. It is also in interest to understand the different users definitions of quality.

To capture the important quality attributes this study has been performed through empirical studies, where an interview has been conducted on the future administrators of the system, and a questionnaire on the other users of the system. The interview's context was at UNN in Tromsø, while the questionnaire was performed on the Internet. The results of these empirical studies were further analyzed and used to test the quality of the KAS. Finally the outcome of the tests and results were discussed, and their validity assessed.

This project is meant to build a fundament for a master thesis were the software system is tested in its original environment at UNN and put into use.

Preface

This report is a product of our work in TDT4735 Software Engineering, Depth Study. This work lasted from August to December 2005 and was a preparation of our graduate thesis of a Master Degree. Primarily all the work have been conducted at the Norwegian University of Science and Technology in Trondheim, except from one trip we had to Tromsø to conduct an interview.

We would like to thank our teaching advisor and supervisor, Tor Stålhane, for all the guidance he has provided throughout the project. We also would like to thank our employees at UNN, Thomas Lyngmo and Jonny Svendsen, for taking of their time to do an in-depth interview with us that gave us a lot of information. A last thank you goes to the employees at UNN who participated in our questionnaire. All answers we received was valuable.

Yngve Halmø *Geir-Arne Jenssen*

Contents

I Background **XV**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Project context . . . 1
1.3 Problem definition . . . 2
1.4 Report Outline . . . 2

2 Prestudy **5**
2.1 The existing system, KAS . . . 5
2.1.1 Initial project background . . . 5
2.1.2 Platform/framework . . . 6
2.1.3 Outline of the KAS . . . 7
2.2 Software Architecture . . . 10
2.2.1 Quality attributes . . . 10
2.3 Software Testing . . . 11
2.3.1 State-of-the-art . . . 12
2.3.2 Testing of Web Applications . . . 13

II Empirical research **15**

3 Research and Planning **17**
3.1 Focus . . . 17
3.2 Methods . . . 17
3.2.1 Qualitative Research . . . 17
3.2.2 Quantitative Research . . . 18
3.2.3 ATAM . . . 20
3.3 Methods of measuring/testing qualities . . . 20
3.3.1 Goal/Question/Metric method . . . 20
3.3.2 ISO/IEC 9126 . . . 22
3.4 Overall work plan . . . 22

4 Empirical Results **25**
4.1 The Interview Session . . . 25
4.1.1 Agreements . . . 25
4.1.2 Quality requirements . . . 26
4.1.3 The contacts definition of usability . . . 27
4.1.4 Quality requirements for other users . . . 27
4.1.5 New functional requirements . . . 27
4.1.6 Other requests . . . 28

4.1.7 Issues that must be addressed . . . 28
4.1.8 Measuring qualities . . . 28
4.1.9 Summary . . . 29
4.2 The questionnaire session . . . 29
4.2.1 Data set reduction . . . 30
4.2.2 Which quality attributes are important? . . . 30
4.2.3 Quality attribute definitions . . . 37
4.2.4 Summary . . . 39

III Software testing 43

5 Quality attribute analysis 45
5.1 Introduction . . . 45
5.2 Attribute utility tree . . . 45
5.3 Analysis of architectural approach . . . 46
5.3.1 "UNN wants to make a change or addition in the code" . . . 47
5.3.2 "User is authenticated at login" . . . 47
5.3.3 "A new user uses the system and manages to place and order correctly" . . . 48
5.4 Sensitivity points . . . 48
5.5 Trade-off points . . . 49
5.6 Risks and non-risks . . . 49
5.7 Risk themes . . . 49
5.8 Conclusion . . . 50

6 Test planning 51
6.1 Goal . . . 51
6.2 Questions . . . 51
6.3 Metrics . . . 52

7 Test results 57
7.1 Metrics . . . 57
7.1.1 M1: KAS precision . . . 57
7.1.2 M2: Error tolerance . . . 57
7.1.3 M3: Wizards . . . 59
7.1.4 M4: Error messages . . . 59
7.1.5 M5: Feedback . . . 59
7.1.6 M6: Modification localization . . . 59
7.1.7 M7: Authorization . . . 60
7.1.8 M8: Authentication . . . 60
7.2 Questions . . . 60
7.2.1 Question 1 . . . 60
7.2.2 Question 2 . . . 60
7.2.3 Question 3 . . . 61
7.3 Goal . . . 61

IV Final results **63**

8 Evaluation and discussion **65**
8.1 Validation 65
8.2 Evaluation 67
8.2.1 Empirical studies 67
8.2.2 Analysis and Testing 69
8.2.3 Important quality attributes and definitions 69
8.2.4 Generally 70

9 Conclusion and further work **71**
9.1 Conclusion 71
9.2 Further work 71

V Appendix **73**

A The questionnaire results **75**
A.1 Computer skills 75
A.2 Important properties 75
A.3 Quality definitions 80
A.4 Thoughts on system 81
A.5 Comments 81
A.5.1 General comments 82
A.6 The questionnaire 83

B Original system requirements **87**

C Relational database schema **91**

D Glossary **93**

List of Figures

2.1 The login screen 7
2.2 The page for ordering permanent id-cards 8
2.3 The start page for admin users 9

3.1 Example of GQM-tree 21
3.2 The project process 24

4.1 The different user groups 30
4.2 The computer skills of the participants 31
4.3 The percentage of each attributes importance for all users 33
4.4 Importance of usability attributes according to "Standard" users 33
4.5 Importance of usability attributes according to "Leader" users 34
4.6 Importance of usability attributes according to "Security" users 34
4.7 Importance of other attributes according to "Standard" users 35
4.8 Importance of other attributes according to "Leader" users 35
4.9 Importance of other attributes according to "Security" users 36
4.10 Users' opinions of usability ordered by user group 38
4.11 Users' opinions of usability ordered by skill level 38
4.12 Users opinions of performance attributes by user type and computer skill 38
4.13 Users opinions of security attributes by user type and computer skill . . . 39
4.14 Users opinions of availability attributes by user type and computer skill . 39

5.1 Utility tree with important scenarios 46
5.2 Scenario 1 47
5.3 Scenario 2 47
5.4 Scenario 3 48
5.5 Sensitivity points 48
5.6 Trade-off points 49
5.7 Risks and non-risks 49

C.1 Conceptual view of the database in MS SQL server 91

List of Tables

6.1 Metric 1, KAS precision 52
6.2 Metric 2, Error tolerance 53
6.3 Metric 3, Wizards 53
6.4 Metric 4, Error messages 53
6.5 Metric 5, Feedback 54
6.6 Metric 6, Modification localization 54
6.7 Metric 7, Authorization 54
6.8 Metric 8, Authentication 55

A.1 E1 - Fast response 76
A.2 E2 - Stability 76
A.3 E3 - Colors 76
A.4 E4 - Appearance 76
A.5 E5 - Well arranged presentation of information 76
A.6 E6 - Able to use without previous knowledge 77
A.7 E7 - Error messages that are easy to understand 77
A.8 E8 - Self explaining tasks 77
A.9 E9 - Difficult to make mistakes 77
A.10 E10 - Protection against hackers 77
A.11 E11 - Protection of personal information 78
A.12 E12 - Enough information to track possible abuse of the system 78
A.13 E13 - Changing password often 78
A.14 E14 - No login 78
A.15 E15 - User manual in paper form 78
A.16 E16 - Search after users and employees 79
A.17 E17 - Show different kind of statistics 79
A.18 E18 - To see the order status 79
A.19 E19 - As many fields as possible should be pre-filled 79
A.20 E20 - Have the option to regret choices 79
A.21 What does usability mean to you? 80
A.22 What does performance mean to you? 80
A.23 What does security mean to you? 80
A.24 What does availability mean to you? 81

Part I

Background

Chapter 1

Introduction

This first chapter is an introduction to our work, where we elaborate the problem and motivation for the project. The last section of this chapter gives an outline of the rest of the report.

1.1 Motivation

In the year 2004 we developed a software system for the University Hospital in North Norway in Tromsø (UNN[1]). The system was developed as a student project at Tromsø University College[2] in the last year of a bachelor degree by a group of four students, including us. The system was an ordering- and administration system for key cards at the hospital. This system, which is described further in chapter 2.1, became more and more complex as the requirements specification grew. The final requirement specification ended up being a lot more extensive than the first version. The final version of this requirements specification can be seen in appendix B. The work is more or less finished according to these requirements, but the system was never tested in the operating environment at the hospital, and therefore never made use of.

The system will from now on be named KAS (Keycard Administration System). The code is about 90% completed, and testing is the main remaining part. The case of the much larger requirement specification may have influenced the system in a less user friendly way and maybe some qualities or requirements have been lost during the development. Both we and UNN want to get the system up and running, because with an electronic system UNN will save a lot of time and money compared to the system they use today, see chapter 2.1.1. This project is supposed to lay the necessary foundation of a master thesis where these goals can be met.

It gives us extra motivation to know that UNN would like to use our system, and this makes us want to deliver a reliable and complete system as possible.

1.2 Project context

This project is carried out at the Department of Computer and Information Science at the Norwegian University of Science and Technology in Trondheim. It is a part of the subject

[1] Universitetssykehuset i Nord-Norge, http://www.unn.no/
[2] http://www.hitos.no/

TDT4735 Software Engineering depth study, and belongs to the Software Engineering Group.

The project is also carried out in cooperation with the University Hospital in Tromsø, which is the final owner of the system when and if it is completed. Our contacts at UNN are Thomas Lyngmo and Jonny Svendsen and they will be the administrators of the KAS. Later in this report when we mention our contacts at UNN, we are referring to Lyngmo and Svendsen.

1.3 Problem definition

If you want to construct a software system of high quality, you need to know which quality properties all the stakeholders[3] consider to be important. The importance of these properties to the different stakeholders can vary greatly, which can be a problem if they are contradictory. When you have established the wishes of the stakeholders, certain questions arise:

- Do they support or contradict each other?
- If they are contradictory, can a compromise be reached?
- In an existing system, does it conform to the stated quality factors or does it have to be changed in some way?

Another problem is that different people can have entirely different definitions of the qualities in question. An obvious example is that a system administrator and an end user can have highly contradicting opinions as to what is user friendly. It could be worthwhile analyzing these differences further.

To summarize, this problem has two main parts. First, we want to establish which qualities and properties the stakeholders find most important in the KAS, and find out how these wishes conform to established quality attribute definitions in modern software engineering, namely those defined by [2] and [1]. In addition, we want to analyze how the different types of users define the following qualities; Usability, Performance, Security and Availability as defined by [2], because these define the properties that we feel are applicable to end-users. Second, we want to analyse these qualities utilized in the KAS to see if there exists any contradictions between them. Furthermore, it is necessary to test the internal parts of the KAS against the most important qualities to ensure that the qualities that were promised the customer in the beginning exists in the system. The external testing (i.e. parts visible to the user) is beyond the scope of this report, as it involves user testing in the hospital environment and this is to be done at a later stage when the internal testing has been evaluated.

1.4 Report Outline

This section describes an outline of the rest of the report. The report is structured the way that the chapters should be read in chronological order, because some chapters provides background for the subsequent chapters.

[3] A stakeholder is any person with an interest in the software system, be it users, investors, corporate heads or developers

Chapter 2, Prestudy, highlights existing work that is of relevance to this project. This includes a brief description of the KAS and an overview of both software architecture and the art of software testing.

Chapter 3, Research and planning, details the research methods used in this project and provides an overview of the overall project process.

Chapter 4, Empirical results, contains the results from the empirical studies. This includes the most important aspects from the interview and the most relevant data from the questionnaire.

Chapter 5, Quality attribute analysis, this is an analysis of the important quality attributes from the empirical studies in chapter 4. Here trade-offs and risks are identified.

Chapter 6, Test planning, provides the ground work for the testing phase with the help of the GQM-method. Describes the overall test goal and it's related questions and metrics.

Chapter 7, Test Results, shows all results from the test phase.

Chapter 8, Evaluation and discussion, contains a data validation discussion of all results obtained in this project and evaluates the important results in detail.

Chapter 9, Conclusion and further work, highlights the conclusions made and details any further work.

Appendix, The appendix contains the complete results from the questionnaire, the original requirement specification, a relational database schema and glossary.

Chapter 2

Prestudy

This chapter provides an overview of the existing system and relevant information in software architecture and software testing. This is meant to provide the background necessary for reading the rest of the report.

2.1 The existing system, KAS

A short introduction to the KAS is in order to provide the basis for any further work. We briefly elaborate the background of the KAS project, and what the system can do and how it has been developed. This is just meant as a summary of the system, and we will not do any in-depth or detailed description of how the system has been implemented.

2.1.1 Initial project background

UNN has many employees, departments and security doors. An elaborate system is in place to control people's access to the different parts and departments of the hospital. Some employees are hired on a long time contract and some are hired temporarily to only work part time. All of these people must have their own key card, to which they use to open doors in the hospital. These key cards only gives access to the doors each employees is supposed to have access to. Some of these cards are lost, others need updating, some most be activated immediately while others are ordered long beforehand.

Every year the security personnel at the hospital handles approximately 800 orders for Id-cards and 2000 orders for time limited key cards. These two types of cards are elaborated later in this section. Today these orders are administrated and handled manually with a lot of paper forms. Every order must be registered by hand and it must be done by a person who has authorization to do so. This system for registering and administrating key cards is both old and cumbersome, and it is easy to mess up the whole system by doing some simple mistakes.

Ordering a card involves a form with personal information that must go through a lot of phases, with up to four persons involved at different times. The "control function" that checks if the cards are delivered on time is relatively complicated and takes a long time to go through. All of this leads to much paperwork and this further leads to relatively much paper laying around to keep history. Further more, users of this system have to deliver orders in person to the security personnel and there exists no mechanism to control that an order is actually delivered, i.e. the orders could disappear in all the paper mess.

This could lead to no card being made, and if this happens people who were supposed to start their work will not get access to the rooms and departments they were supposed to. Thus, disappearing of orders must be prevented at all costs.

There exists two kinds of key cards and therefore two kinds of orders forms, those that go to employees working full time (id-cards) and those that go to people only working part time (time limited cards). The part time cards are only issued to people for a short period of time. Employees working full time get their own Id-card with their picture on it. People with cards gets access to all the doors they are supposed to in their department which they work at. When a job is done or a persons quits his/her job, all key cards have to be delivered back to the security personnel. It is especially important that no one gets access to a door or unit they not are supposed to. Just imagine what could happen if someone, who is unauthorized, gets access to a medicine room. Security is, as one can see, a crucial part of the system. The security personnel must at all time know how many cards that are in the system, who has these cards, when these cards should be delivered, and last but not least, be sure that only authorized personnel can order the key cards to other people.

Our assignment at Tromsø University College was to develop an electronic system for ordering key cards at UNN that could replace the old paper system described above. The electronic system would save UNN a lot of time and money by enhancing and lightening the work done with these orders. The implementation of the system was finished according to the original quality requirements, but the hospital couldn't take it into use at the time we were finished with it, as it had to be tested to ensure that it had all the promised functions and that the users could use it effectively.

2.1.2 Platform/framework

The KAS is developed as a web interface. The interface is reachable with any Internet browser from any PC at the hospital. This is a good thing because it makes it unnecessary to install any programs on the computer before using it. The program, or system itself, is located on a server which is accessible within the intranet at the hospital only. This reduces the possibilities of hacking from the outside compared to if the system was accessible from the Internet.

The framework used to develop this system is ASP.NET [1]. It allows you to access .NET Framework Class Library that offers thousands of classes with all kinds of services. ASP.NET is a scripting language which is processed at the server side. When a client sends a request of some webpage with ASP code the server processes the page and sends the output to the client in form of a HTML[2] page. ASP.NET-files consists of both HTML and ASP-code in some programming language.

HTML is the standard programming language for making webpages. It can be displayed on different operating systems and in different browsers. HTML-files is simple text files with "tags" defining the structure of the page, format, pictures, links and so on.

[1] ASP.NET is a web technology that is developed by Microsoft. ASP stand for Active Server Pages, see http://www.asp.net

[2] HyperText Markup Language, the standard for making web pages. Utilizes special "tags" to describe the formatting of a web page. For more info see http://www.w3.org

The whole site is connected to a database that stores information and data. This database is a Microsoft SQL database[3]. The database is the most complex part of the KAS, but we will not go into any detailed description of it. Any modification of this database would be very difficult and will at least require extensive knowledge of the enitre system. To control the flow of data through the system a lot of constraints and relationships has been constructed, see the relational database schema in appendix C.

2.1.3 Outline of the KAS

What you see when you open the KAS is a login page. This login page can be seen in figure 2.1. If you don't have a username and password, you can't log in to the system. This is to ensure that only authorized personnel is able to submit orders. The system is built with 5 different user levels. That is "ekstravakt", "avdelingsleder", "idkort", "vekter" and "admin". The user levels are ordered in kind of a hierarchy, with the highest level having all the functions of the lowest levels. After logging in, the system provides different options depending on what kind of user one are. The user levels will be described further and the description begins with the lowest level and increasing towards the highest.

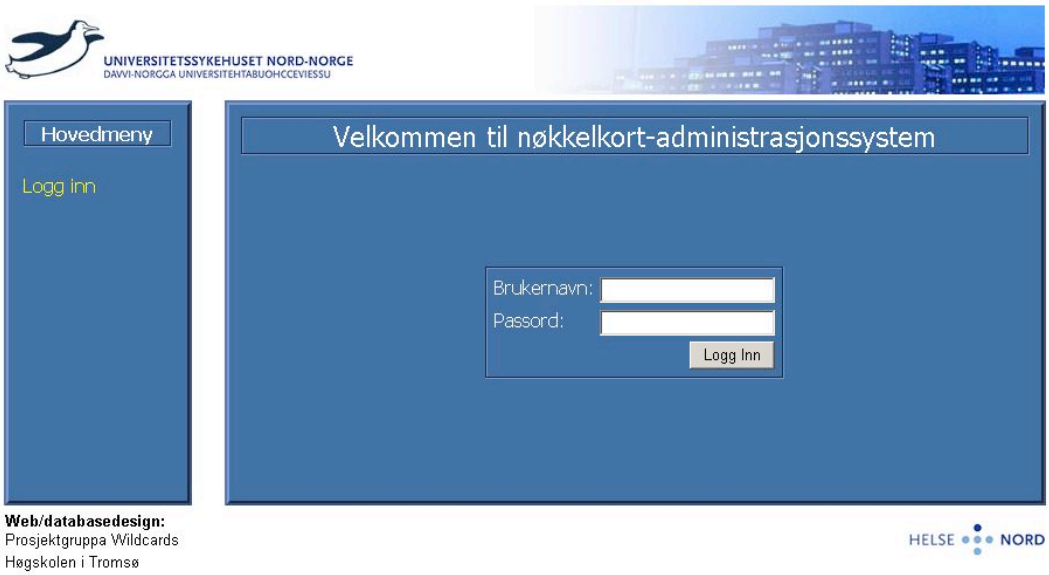

Figure 2.1: The login screen

Ekstravakt (Extra shifts)

Users with this user level is able to order key cards to employees in their own department who work part time, and see status of these orders (i.e. if the order is in queue, under treatment or done). They are able to send messages along with the order to the security personnel if they want, and they are able to see answers to these messages. This user level is the lowest level in the system, and the only available function except submitting these orders are the ability of changing their own password.

[3]Microsofts implementation of the SQL database standard, www.microsoft.com

Avdelingsleder (leader)

This user level has the same function as the "ekstravakt" plus the opportunity to register new users with this privilege.

Id-kort (id-cards)

This level is the third lowest level in the system, and users with this level has all functions of the "ekstravakt"-level but is also able to order id-cards to employees. Id-cards is a card with a picture of the owner, and these cards go to employees on a long time contract. The page for ordering id-cards can be seen in figure 2.2. It is basically only the personnel department (personalavdelingen) that has this user level. They can order cards to any department.

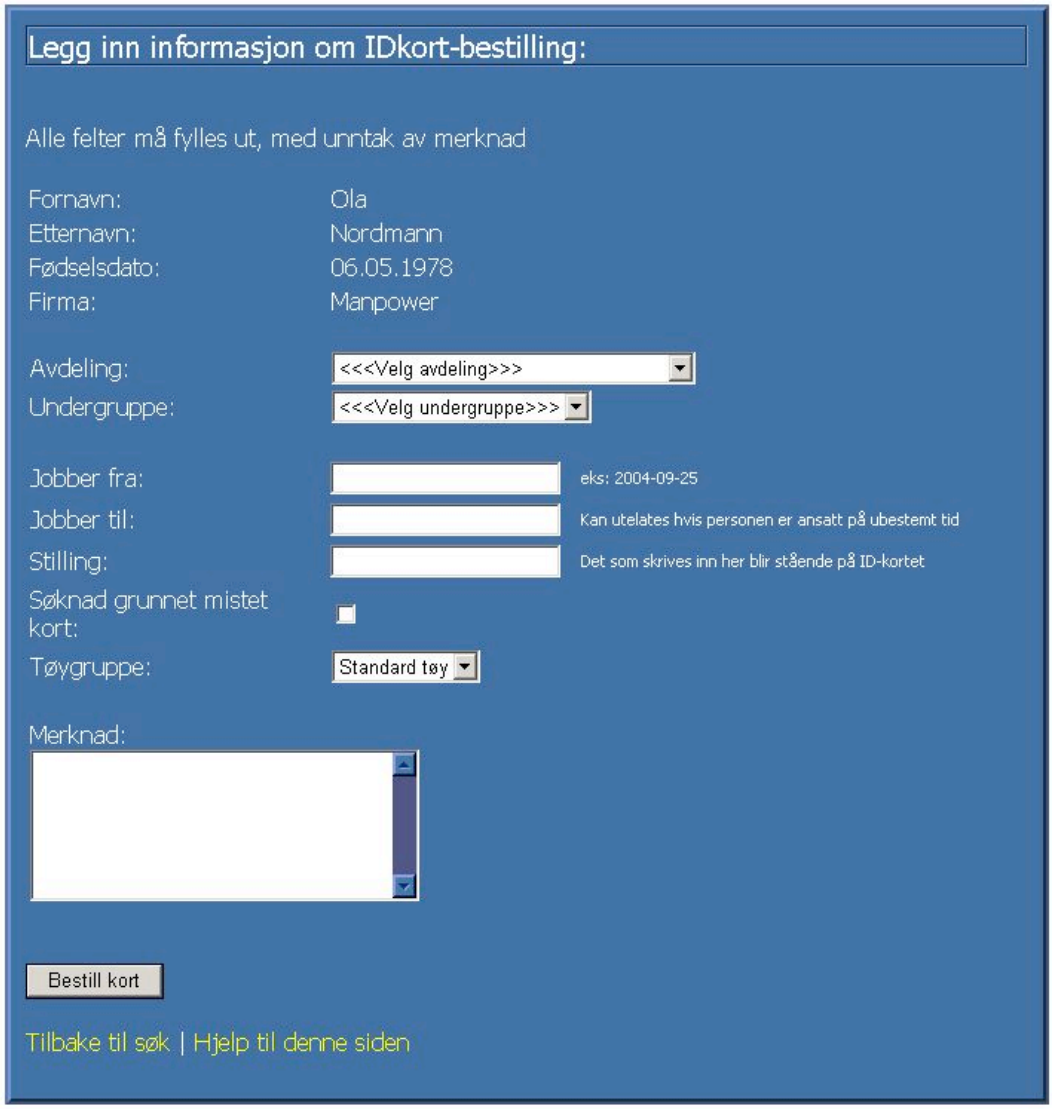

Figure 2.2: The page for ordering permanent id-cards

Vekter (security personnel)

This user level is significantly different from the previous. These users will receive all submitted orders and activate key cards[4] as they approve these orders. This user level has several useful functions, like the ability to inspect all logs, i.e. which cards should be delivered, which are not delivered, who has what kind of card and which cards that must be returned in the near future. In addition this user group has access to all functions of the lower user levels, like submitting an order, changing passwords etc.

Admin

"Admin" users have all rights and has access to all functions in the whole system. This includes the ability to update several tables in the database easily (i.e. delete a department, insert a new department, delete and insert access levels that is used on key cards) with few clicks in the system, and decide which other people that should have access to this user level. The homepage for Admin users is shown in figure 2.3.

Figure 2.3: The start page for admin users

[4]This is done on a different, external system

2.2 Software Architecture

"Requirements beget design, design begets system" [2]

This is the traditional approach to software designing, and although newer software development methods have improved on it, most of them still make the assumption that design is exclusively a product of a system's technical requirements [2]. Architecture has emerged as an abstract way of designing software, and can be defined as follows: "The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them" [2]. Other definitions include "Architecture is high-level design", "Architecture is the structure of the components of a program or system, their interrelationships, and the principles and guidelines governing their design and evolution over time" [2]. In any case, a software architecture gives a software development process methods to assess risks associated with launching unprecedented designs, and techniques for uncovering design flaws before it is too late to correct them. While a complete architecture can be very comprehensive, we intend to concentrate on the parts of it that highlight how the different stakeholder's wishes interact with each other, namely defining quality attributes and evaluating them with the ATAM[5]. A CBAM[6] could have been interesting in a normal business environment, but because of the educational environment in which this project exists, it does not seem applicable. It can, however, be worth looking at in the future, at a later stage in the system's development process.

2.2.1 Quality attributes

The meaning of quality is often in the eye of the beholder and can be defined in several ways. It can also be depending on a specific situation. To classify a computer system's quality one could take a look at the architecture and it's properties, and create different "parts" depending on their properties. One way to do this is to divide the computer system into quality attributes. Quality attributes is a way of categorizing the requirements of a system, not only the functional ones but also how code should be structured, how safe it should be and how user friendly it should be, to name a few. [2] defines quality categories as:

- Usability, concerned with how easy it is for a user to accomplish a desired task and the kind of user support the system provides
- Availability, concerned with system failure and it's associated consequences
- Performance, concerned with timing and how the system responds to events like interrupts, messages, requests from users and passage of time
- Testability, refers to the ease with which software can be made to demonstrate its faults through testing
- Modifiability, concerned with the cost of change, what aspects of the system can change, when changes could be made, who could make them and how much time and money they will cost

[5] Architecture Tradeoff Analyses Method, see chapter 3.2.3

[6] Cost-Benefit Analyses Method, A method of analyzing future costs and benefits associated with a software system

- Security, a measure of the system's ability to resist unauthorized usage while still providing its services to legitimate users

[2] also recognizes that scalability, portability and interoperability are being used as other quality attributes, but feel that they have largely incorporated them into their own definitions listed above.

A collection of ISO[7] standards exists for software engineering[8], and they define the quality attributes of a computer system as follows:

- Functionality, the capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions
- Reliability, the capability of the software to maintain the level of performance of the system when used under specified conditions
- Usability, the capability of the software to be understood, learned, used and liked by the user, when used under specified conditions
- Efficiency, the capability of the software to provide the required performance relative to the amount of resources used, under stated conditions
- Maintainability, the capability of the software to be modified
- Portability, The capability of the software to be transferred from one environment to another

For a description of some of the terms used, see the glossary in Appendix D.

2.3 Software Testing

Testing is an essential part of any software development. Even though a program does what it is intended to do, it might still be a possibility that there exists errors that makes the program do things it is not supposed to do. A test can only prove that an error is present, not that an error is not present [5]. A successful test is one that finds errors, not one that doesn't find them. It is therefore essential to test new software to ensure high quality.

According to Hutcheson [4] software testing today seems to have four main problems:

- *Ways of development:* Many new ways of developing software are based on trial and error methods. With these methods there exists no specifications to test against, so the testers are left to hunt for bugs.
- *Schedule:* In the recent years development have been driven by constantly evolving product definition and management is not always convinced that testing is worthwhile because they want their product on the market before any competing company and product.
- *Few using formal methods:* Few testers are using formal methods and metrics, and as a result of this the test effort is not producing the kind of quality results that improves the quality of the product and the bottom line.

[7] International Standardization Organization
[8] ISO/IEC 9126-1, 9126-2, 9126-3 and 9126-4

- *Standardization:* The most important software quality improvements in the past several years has come from standardization driven by the Internet. These standards have had an effect on the ability of multiple software systems to interact and interoperate, this has removed the need for extensive low-level testing.

2.3.1 State-of-the-art

Modern software testing have in some respects come a long way since people started writing software, however, in other ways the evolution has been almost stagnant. New, advanced testing tools have been created that make testing easier. Pre-tested subroutines exist that can be used right out of the box. Development environments and compilers identify a large quantity of errors, still, the percentage of resources required to properly test a software system is about 50% of it's total development cost, as it has been since the early 80's [5]. Increasingly powerful computers, many new operating systems and programming languages, and a plethora of possible hardware combinations have complicated the matter. Computer programs also continuously grow in size, making the task daunting.

"Testing is the process of executing a program with the intent of finding errors", according to [5]. They assert that one of the primary causes of poor program testing is the fact that most programmers begin with a false definition of the term, for example "Testing is the process of demonstrating that errors are not present". This implies that the technical nature of software testing is actually influenced by human psychology. It also implies that you should not test a software system you have created yourself, because you subconsciously will look for errors where you are less likely to find them since it is your own code. Getting someone else to test our software is however out of the scope of this project, so we must take care not to step in this trap. However, according to our supervisor Tor Stålhane, the opposite definition of testing is true, namely that testing is to show that we have delivered a product as promised. In any case, both angles should be considered.

Two important testing strategies are black-box and white-box testing. The first views the program to be tested as a black box, i.e. something which internal structure you know nothing about, but you do know it's interface and the functionality (from the specifications). This strategy consists of input and output values. If the input values, when compared to expected output values, match, the test will pass. If they don't match it will fail. If one is supposed to find all errors, a huge number of input and output values must be provided. White-box testing permits you to examine the internal structure of the program, and try to make test data that, for example, tests every program statement at least once. None of these methods are good enough on their own, mostly because testing every possible combination of input data or path through the program logic is nigh impossible, and even if you managed to do it, there would probably still be errors you did not identify. Another, more interactive strategy is human code inspection. By using a checklist of known programming errors you can root out a lot of bugs before you even write a test case. A combination of black/white-box techniques and non-computer based testing procedures is recommended.

As the size of a software system grows, the number of design errors relative to code errors tend to rise [5], and higher-level error-testing becomes more important. In this project we will have to consider both levels.

2.3.2 Testing of Web Applications

Applications on the Internet introduce new challenges in the area of testing [4]. These challenges consists both of the architecture of the web applications, the results of users using different web browsers and the high focus on user interface the applications have. A user of a web application interacts with the system through a server sending him or her responses to his or her requests. The server processes all application data before the client shows the results to the user.

Different layering of web applications make testing of these applications harder. Several servers uses data stored in databases, and this data consists not only of the content shown on the web pages but also the security constraints related to a users login (in applications with such login system). As one can see, a lot of things complicates the testing process and some components couldn't be tested alone.

Client/server systems have to be tested both at the server side and the client side. There exists some difficulties with both of these tests. With server testing it could be difficult to do unit testing. To test if a component works as expected in a system with database could depend on a lot of factors. For example to test if a user has authorization to one part of a system but not to another part depends if the user is registered in the database. Also the order of which each test is performed could pose difficulties. Two or more tests could for example have an indirect impact on each other with the first test changing the system (or the way the system behaves) so that the second test is affected. This is not a good thing as the second test will respond differently depending on the first test. Client testing could be to press buttons, fill out forms, check links and read text in different browsers to ensure that the system is displayed properly to all the users. In some browsers the layout of the pages could be displayed wrong, and this is something that one should avoid in order to make the system more user friendly. This kind of testing is time-consuming, and doing it manually makes it vulnerable to human errors like overlook spelling errors, or pressing a button one is not supposed to. This could lead to even more errors.

Even though the KAS is not supposed to be accessible from the Internet, it is in fact a web-based application and therefore it may have to be tested as an web application.

Part II

Empirical research

Chapter 3

Research and Planning

This chapter describes the research methods used throughout the project. It also outlines the plans for the project process.

3.1 Focus

In order to capture the future user's opinions and preferences we need to do some research. Therefore, we plan on doing some empirical studies on these users. A more detailed description of these studies can be found in the following sections. When the users have conveyed their wishes we plan to use these results in an analysis of the most important qualities of the KAS, to see if any trade-offs have to be made between them, and if the software needs to be changed. To do this, we see an ATAM analysis as a good choice. To investigate if the given quality requirements are met we then plan to conduct low-level internal testing of the software. At the same time, via the empirical studies, we will do some more general research in software development, namely try to determine any differences in how different users define quality attributes.

3.2 Methods

In empirical studies there exists two kinds of research methods; qualitative and quantitative research. Although these methods may be somewhat opposing in the way they gather data and what kind of data they gather, they should be seen as complementary to each other rather than competing in order to get the best results. We have chosen one of each of these methods to meet our goal in this project and will describe how we are going to use them in the following.

3.2.1 Qualitative Research

Qualitative research methods try to recognize every single object(s) or person(s) that is under study[1]. The goal is to understand the world as seen from the subjects point of view. These methods are characterized by being subjective, open and profound. Two examples of qualitative methods is interviews and observations. Interviews of subjects could be performed to get valuable information. One obvious use of the observation method could be to observe some users during a user testing session were they provided us with feedback.

[1]The person or object under study is often referred to as the subject

The advantages of doing interviews are several, as opposed to doing questionnaires on paper. You get more feedback during an interview session, and you can observe how the other party reacts to a certain question, and act accordingly. The interviewer is able to reduce the number of "I don't know"-answers because he or she is able to explain the true meaning of a question should it be unclear. The interviewer is also able to ask follow-up questions that may emerge naturally during the session, while a questionnaire would not provide this ability. Disadvantages of an interview includes the costs involved in having an interviewer present, and the prolonged time required compared to questionnaires which can be answered simultaneously in unlimited number. The quality of the answers you get in an interview will to some degree depend on how good the interviewer is, be it better or worse.

Interview

By request from our supervisor, we decided to conduct an interview with our contacts at UNN in person, because of the advantages of this research method. While a questionnaire could have been conducted on our contacts instead, an interview was preferable for several reasons. First, being the contacts, their wishes concerning the software system we are developing are paramount, and should therefore be documented in detail. While the quality requirements already exist, the project has been on hold for a year, and therefore new issues need to be resolved. Second, they will also act as the administrators of the system once it is operational, and being the only two representatives from this group of users, a more qualitative approach seemed necessary. Third, they can also act as security personnel, alleviating the need for interviewing those users qualitatively. And finally, we know from previous experience that they are technically competent, and have much experience with the existing paper system and the KAS, and can therefore provide invaluable insight to the development process.

The questions in the interview were deliberately made open, to support loose discussion. The scope was set to 1 hour, and it was decided that it should be recorded by a voice recorder, to avoid missing any details. The results of the interview is presented in chapter 4.1.

3.2.2 Quantitative Research

Simply put, quantitative research is about numbers. Analysis is often based on counting and comparing the results. A calculated sample of a population is asked a set of questions to determine the frequency of their responses. This method is often more objective than the qualitative method, and also more structured and less flexible than qualitative methods. Two examples of quantitative methods are questionnaires and experiments. In experiments one controls the input and the goal is to observe some output. This method is not suitable for this project, and therefore we will not discuss experiments further.
A questionnaire is often built up of general questions that all people in a population could have an opinion about, that be either positive, negative or neutral. Questionnaires could for example be done to perform an opinion poll, either to register people's opinions about an existing product or case, or their expectations of a new one. Getting answers from several people rather than one or two makes generalizing the answers to even a larger population easier. There exists a lot of statistical analysis and methods to aid in

this case. Every aspect or question in a questionnaire can be quantified and made into some statistics. The margin of error and risk of error must, however, be carefully considered and validation of data is therefore important.

Wohlin [7] says the general objectives for conducting a survey is either descriptive, explanatory or explorative. Of this classification our objective would be descriptive since we are interested in enabling assertions, as Wohlin explains, about some population. We want to examine the understandings of certain quality attributes, and therefore "the what" rather than "the why" is important in this setting.

Questionnaire

We have chosen to perform an Internet-based questionnaire to register the opinions of the different users of the system. We could think of several advantages that made the decision of making the questionnaire on the Internet easier:

- **Speed** You could distribute the questionnaire quickly, and get answers shortly after the distribution.
- **Statistic** If you connect it to a database, you could easily make any statistics of all the answers.
- **Costs** The average costs of doing a survey with many people would be lower than distributing it on paper. Then you have to print out and mail the survey, before waiting to get the answers returned by mail to us. This will take time.
- **Modify** If the first few answers received suggests that additional questions should be asked, this easily can be modified. To modify this by ordinary mail would be difficult.
- **Attractive** The survey could be made attractive with graphics and fonts. It is also possible to add video or audio to make it even more interesting.
- **Flexible** The population can answer the survey at any time up to the deadline, not only immediately when they receive it (as it is with telephone surveys).

We discovered only two main disadvantages with internet questionnaire. First, it took a considerable amount of time to make the web page with the questions and then connect it to a database. To be certain that it would not suffer any unforeseen technical difficulties due to external issues, we hard coded it from scratch. Second, since the questionnaire would be located on a public Internet server, open for everyone to answer, we could receive data that is corrupt and not from the population we are asking. We could have circumvented this by adding authentication like passwords to be distributed in the target population, but feared that this would decrease the number of answers due to the heightened difficulty and hassle for the user. We ultimately concluded that this would only be a minor problem, since the page address was not publically available, limiting chance hits, and should someone find it, it is unlikely they would actually bother to answer it. In addition, by reading through the submitted data, garbled answers or answers of a less serious nature would be easily identified.

The questions from this survey can be seen in appendix A and the results will be handled in chapter 4.2.

3.2.3 ATAM

The Architecture Tradeoff Analysis Method (ATAM) [2] is a way to reveal how well an architecture satisfies particular quality goals, and it provides insight into how quality goals interact, and trade off. It does this because it recognizes that architectural decision tend to affect more than one quality attribute, often adversely. The analysis produces several outputs, and the ones we are interested in are listed here:

- Quality requirements in terms of a collection of scenarios
- Mapping of architectural decisions to quality requirements. Architectural decisions can be interpreted in terms of the qualities they support or hinder. For each quality scenario examined during the ATAM, those architectural decisions that help to achieve it are determined
- A set of sensitivity and tradeoff points. These are architectural decisions that have a marked effect on one or more quality attributes. A sensitivity point is an architectural decision that positively affects a quality attribute. If a sensitivity point adversely affects another quality attribute, it is identified as a tradeoff point
- A set of risks and non-risks. A risk is an architectural decision that may lead to undesirable consequences in light of the stated quality attribute requirements. A non-risk is one that is deemed safe
- Risk themes tries to summarize the risks and to look for weaknesses in the architecture. If one or more emerge they should be taken seriously

3.3 Methods of measuring/testing qualities

"What is not measurable make measurable" - Galileo Galilei(1564-1642) -

"In software engineering it is difficult to define an attribute in a measurable way which everyone agrees on." [4]. With this in mind we plan to structure the test phase using the GQM method, and combine it with some recommended internal tests from the ISO 9126 standard for product quality. Both methods are described later in this section. If every part of the system were to be tested, including every possible input and every possible logical path it would take virtually forever, so a selection will have to be made. But by using the methods mentioned, we should get a structured way to conduct the tests that combines elements from black-box, white-box and human testing.

3.3.1 Goal/Question/Metric method

The Goal/Question/Metric (GQM) method is a systematic approach to build research questions in a structured way. This method was a result of V.Basili and D.Weis work from both practical experience and academic research [3].

The basic idea is to derive software measures from measurement questions and goals. The method is built up in a hierarchical way with three levels. This hierarchical structure is shown in figure 3.1. At the highest level the goal is located, which is the purpose of the measurements. Following the goal, at the middle level, is the questions related to the goal. These questions characterize the way the achievement of a specific goal is going to

be performed. At the third and lowest level is the metrics, associated with the questions, located. These metrics defines a measurable way in order to answer the questions.

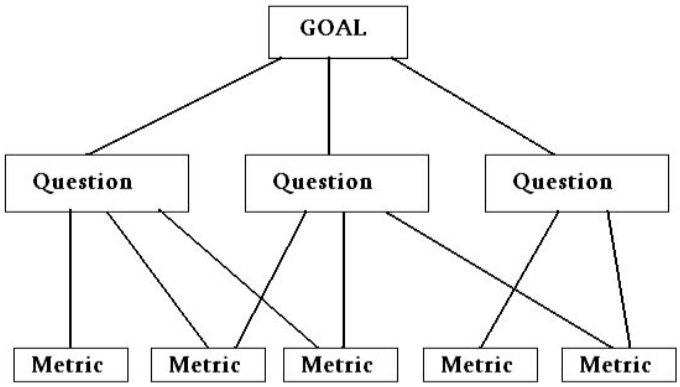

Figure 3.1: Example of GQM-tree

The method consist of four different phases as described in [3]:

- **The Planning phase**, where the project is selected, defined, characterised, and planned, resulting a project plan.
- **Definition**, where the goal, questions, metrics and hypotheses are defined and documented.
- **Data collection**, where the actual data collection takes place.
- **Interpretation**, where the collected data is processed with respect to the defined metrics into measurement results. This will provide answers to the questions, which is needed to evaluate the goal.

Goal

This section describes the goal of applying the method. A project might have several goals. The method uses a standard template to defining its goal according to Wohlin [7]:

> Analyze <*Object(s) of study*>
> for the purpose of <*Purpose*>
> with respect to their <*Quality focus*>
> from the point of view of the <*Perspective*>
> in the context of <*Context*>

Questions

Questions must contribute to the goal and should not be possible to answer with a simple yes or no.

Metrics

The metrics answer the questions by measuring different aspects. The metrics can be answered either objectively or subjectively, and there can be several metrics that contributes to one question.

Our goal/questions/metrics are presented in chapter 6. The results from the data collection and interpretation are presented in chapter 7.

3.3.2 ISO/IEC 9126

ISO[2] and IEC[3] form the specialized system for worldwide standardization. ISO and IEC have established a joint technical committee ISO/IEC JTC 1, which have created a framework for software quality evaluation because "As software applications have grown, so too has the importance of software quality. In order to specify and evaluate software product quality objectively and quantitatively, a framework for software quality is necessary." [1].

The ISO 9126 standard is divided in four parts;

- 9126-1 is an overview and hierarchical categorization of quality characteristics into six main categories and their respective sub-categories, such a category is called a quality attribute
- 9126-2, 9126-3 and 9126-4 are technical reports on, respectively, external, internal and in-use metrics for measuring the quality attributes defined in part 1

In this project, we will concentrate on "internal metrics", as defined by 9126-3, and utilize these metrics for testing. External metrics largely concern user testing, which is considered future work and outside the scope of this project. We will define quality attributes according to [2], and not according to this ISO standard. The definitions are quite similar and basically describe the same aspects with different words. We chose the former because of our experience with the terminology.

3.4 Overall work plan

The activities/phases done during this project can be seen in figure 3.2. The time limit is from start at the top to delivery at the bottom, and most of the activities depends on the completion of the above activities. Only those placed side-by-side we have been able to perform at the same time, independent of each other. The figure only shows by which order we have performed all the activities, not how long each activity lasted. The different activities will now be described further.

Start: The project starts with a discussion of possible projects, and acceptance of this project by our teaching supervisor.
Problem Definition: In this phase we defined the problem at hand, and found a suitable task which both NTNU and our contacts at UNN was happy with.

[2] the International Organization for Standardization
[3] the International Electrotechnical Commission

Report: Denotes the actual writing of this report. This is an activity which has been ongoing from the problem definition through to the delivery.
Prestudy: This is a study of existing technologies and materials needed. We knew a lot beforehand but needed to find out a few new things, specially about testing where we had little experience.
Project planning: This is the planning phase were we planned all the work that had to be done to answer the problem at hand.
Interview session: This phase includes also the preparation of the interview. We were in Tromsø at UNN to conduct an interview with our contacts. This was an important aspect of the whole project, we had to clarify some issues and they provided valuable information.
Creating questions for questionnaire: At the same time when we prepared for the interview session, and conducted the interview, we began creating questions for the questionnaire. These questions was partly formed from the interview questions, and partly from our knowledge of the KAS.
Implementation of the Internet questionnaire: After the Interview, we immediately started the implementation of the questionnaire as a webpage. It turned out to be harder than we had imagined, as we hard coded it from scratch and it had many data fields that had to be connected to a database.
Analysis of empirical studies: In this phase we analysed the result both from the interview and the questionnaire. We had the interview recorded on a voice recorder, and this made it easy to pull out the most important factors. It was a bit worse with the questionnaire though, as we was unsure of which data representation we should use. It was relatively easy to create statistics on the computer, but harder to choose among the enormous amounts of possible presentations and categorizations we could use.
Trade-off analysis: To answer the part of the problem definition which concerned the conformation or contradiction of the quality attributes, we chose to perform an Architecture Tradeoff Analysis. We needed to retrieve the most important quality attributes from both the interview and the questionnaire. This was done at the same time as the analysis of the empirical studies, right after the most important attributes from this studies were ready.
Test planning: The starting point of the test planning was the results made from the analysis in the above activities. We wanted to test the system to see if it already had the qualities required and if not, what must be done to correct it. We used the GQM-method to guide us in the planning of the testing. By doing this we got a set of questions that had to be answered and a set of metrics to answer them.
Testing: Performing the metrics from the GQM made us test some internal parts of the system.
Test results analysis: Analysis of the metrics in the GQM.
Validation: Along with the discussion, we had to validate the methods and results to be sure that we could draw a conclusion that was valid and trustable.
Discussion of results: This is the discussion phase, where we evaluated the whole project.
Delivery: The final delivery of the project report to NTNU.

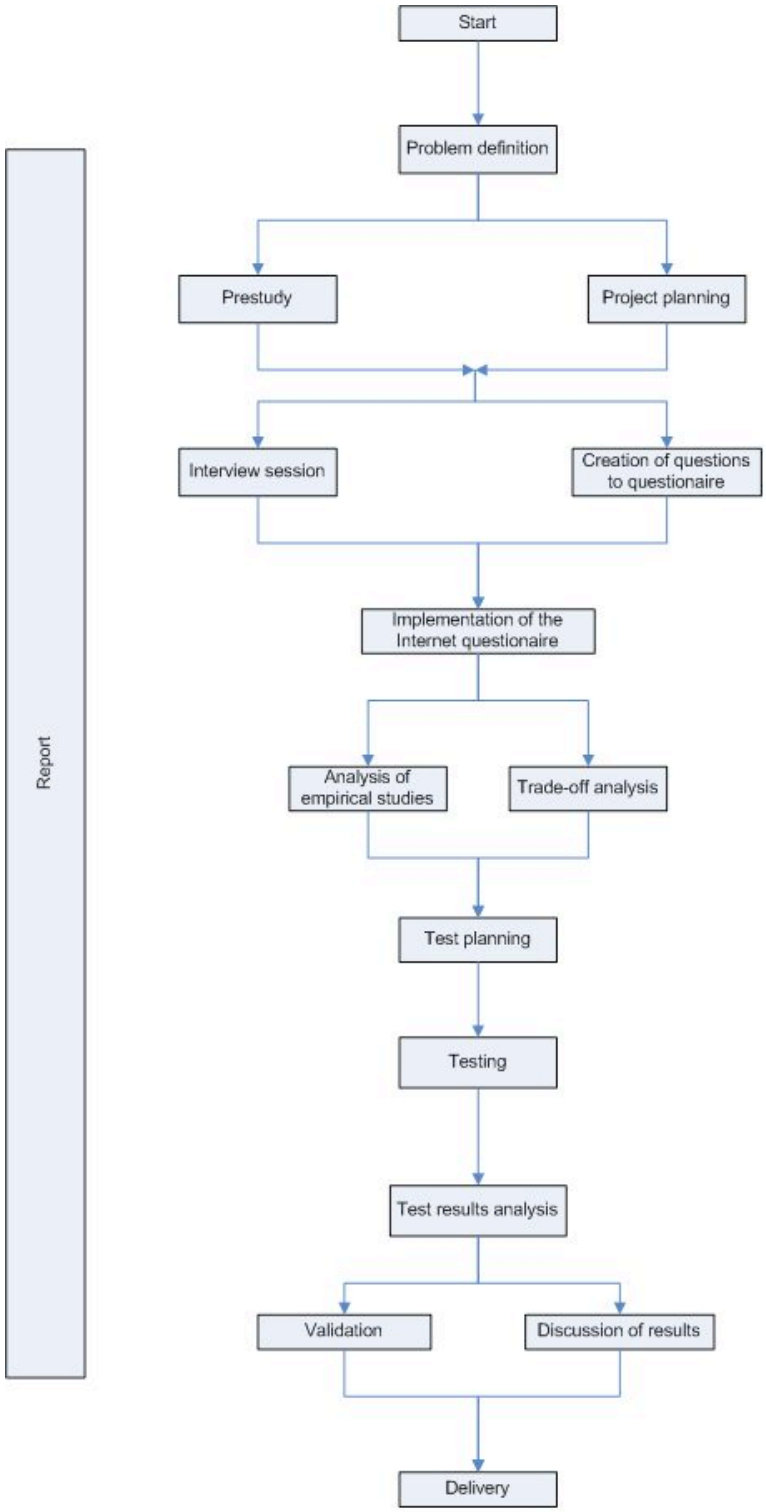

Figure 3.2: The project process

Chapter 4

Empirical Results

In this chapter we will present the results from the empirical studies, namely the interview and the questionnaire. Only the most relevant data from the questionnaire will be presented here, the rest can be found in appendix A.

4.1 The Interview Session

This section will sum up the significant results from the interview in Tromsø. The subject on this interview was our contacts at UNN and the whole session lasted for approximately 2 hours.

4.1.1 Agreements

First we talked about some formal issues, and the most important are listed in the following:

- The IT-department of the hospital will be responsible for the system administration, specifically ensuring that the web/database servers are operational. They will also provide backup-functionality. The contacts will provide us with a liaison in this department.

- The contacts will make sure that enough users complete the questionnaire when it is ready.

- The contacts will make the user test phase a priority, and divulge the necessary resources, in terms of time and personnel. The system to be tested is the completed version based on the existing quality requirement document, however we are willing to include new functionality while time permits.

- We discussed the possibilities of a simulated test phase, as realistic as possible, on a selected group of users where they will be subjected to known issues. The scope will be one day. This will reduce the risk of integrating the system in normal procedures, and should produce valuable insight and evaluation of the system. Both parties shall strive to complete this phase as early as possible.

- The primary goal is that the system shall be operational before summer of 2006.

- All extra functionality has less priority than getting the system operational

4.1.2 Quality requirements

Further we discussed what properties they considered to be important in the KAS, and the most significant aspects is classified by quality attribute and presented here. From some of the descriptions of important properties their understanding of the respective attribute could be deduced.

Security

- Primarily concerns authentication, ensuring that orders originate from the right person/department, and that it concerns the right person/department.
- Authentication with signatures is an important part of the existing paper-based system, and it is important that the software system incorporates an equally high level of security.
- Passwords must be kept private. The biggest risk concerning this is probably not in the software, but in the users behaviour.
- The number of security loopholes must be minimized. And those that exist must be handled by satisfactory routines.
- The system shall only be available on the hospitals intranet, inside the firewalls.

Usability

- An important property is precision. By precision it is meant that there must be no uncertainty about the processes that the system shall be used for. All tasks the user must do can not be prone to user errors.
- The graphical user interface must be intuitive and easy to use, preferably self-explanatory.
- The threshold the users must cross to familiarize themselves with the system must be low.
- Trade-off with safety. Usability is very important, but security is paramount.

Availability

- Downtime requirements: 1 day is acceptable, 3 days is not.[1]
- The system will be protected by existing failsafe routines that protect more critical systems, which should be satisfactory.
- Redundancy will be handled by keeping the existing paper-based system as backup in the early life of the system. The existing paper system will have to be used in some situations in the early implementations of the new system anyway.

Portability

This aspect is less important, because it is highly unlikely that the system will need to be ported to any other format or system.

[1] Continuous downtime is the worst, short periods are acceptable

Modifiability

This is important, it HAS to be possible for people other than the existing developers to make changes to the system/add functionality. Intuitively, doing this must be cheaper than creating a new system.

Performance

The minimum requirement is that the response time does not become so high that the users start making mistakes because of it. However the system's demands for network traffic and server load will be far below the server cluster's existing capabilities, so it should not become a problem.

4.1.3 The contacts definition of usability

- That the system is intuitive
- That it doesn't require large manuals or lots of training to learn
- Buttons are self-explanatory
- Simple things like how you name an item are important
- It must be possible to read from the active screen what it is you are supposed to do
- Pop-up windows that you need to switch between are not very user friendly
- Overview must not be lost

4.1.4 Quality requirements for other users

Here the contacts were asked which features they thought would be most important for the other users of the system.

- Be able to check an orders status, to determine why a card doesn't work, or why it never arrived etc.
- Few and simple operations are needed to complete an order, as many fields as possible should be pre-filled.

4.1.5 New functional requirements

- If the hospital expands the number of physical units that accept incoming orders, the software must be changed to ensure that the correct unit receives the order. The department-table might get an extra column which links all the departments to their respective units.
- The KAS should ultimately be integrated with the access control system, so that when an order is accepted the card's access parameters could be automatically updated. This is, however, the lowest priority at the moment.
- Control over cards that should have been returned to the administration should be improved. Better mechanisms are needed for this to work in practice, although this does not necessarily concern the KAS.

- More levels of security in the authentication process is desirable. This may include logging NT-user info and computer name/IP address.

- Functions for generating arbitrary statistics is desirable

- If errors occur and the user gets stuck, contact information must be provided to solve the problem.

4.1.6 Other requests

- Technical documentation of the system has high priority, and some educational aids are desirable

- The questionnaire should enable the users to point out the biggest flaws of the existing system.

4.1.7 Issues that must be addressed

- The hospital has introduced a new management system for employees, and including the access control system and the KAS this makes 3 individual databases containing information about employees/cardholders. The interconnection(s) between these must be discussed, primarily how they shall be kept up to date.

- Using social security numbers for database identification will be problematic if not impossible in practice, and must be addressed. The KAS, however, does not use this kind of identification.

4.1.8 Measuring qualities

The employers were asked if they could think of adequate ways to measure the quality requirements in question

- A way of measuring usability is counting how many orders the security personnel has to manually enter into the system. This percentage of the total number of orders received will shed light on the extent to which the system is used effectively. If the percentage diminishes over time, the KAS should at least be considered an improvement. The KAS could for example log the total number of orders received over a period of time and the number of orders submitted by security.

- A member of the IT-department could review the code and give a qualified opinion about how it conforms to the quality requirements, for example if the code is modifiable, well commented, or generally lives up the department standards. Such reviews could be conducted at any time.

- Using test subjects with no prior experience with the system could indicate how intuitive the system is.

- Users can, in general, determine if the system fulfills its quality requirements

4.1.9 Summary

A short summary of the most important aspects from the interview:

- Security - Specifically authentication and authorization are very important aspects. The system must ensure that people are who they say they are, and that they are given the appropriate access. The level of security and traceability provided by normal signatures in the paper-based system must be ported to the software system.
- Usability - Precision. The users must have as few options, or ways of doing things wrong, as possible. The system should use dropdown lists instead of free text fields where possible, remember previously entered information and tell the user in simple terms what he/she is doing wrong whenever this occurs. This will both make it easier to use, reduce the number of errors, and increase security.
- Modifiability - It should be as easy as possible to modify/expand the system without the help of the developers, UNN does not want to be totally dependent on the developers. Extensive technical documentation is important with respect to this point.

The contacts definition of usability is that a system must be intuitive and self-explanatory. It must also be easy to maintain an overview of what you are doing and where you are in the system.

Some new functional requirements compared to the initial requirements specification have been mentioned, but it has been made clear that anything that can prevent the system from becoming operational does not have priority, this is the primary goal at this time.

4.2 The questionnaire session

The questionnaire was, as described in chapter 3.2.2, conducted on the Internet. The questions were drafted, then evaluated through brainstorming, experience and our knowledge of the KAS. These questions along with the complete results of the questionnaire can be found in appendix A. A webpage containing the questions was created in HTML and PHP[2]. A MySQL[3] database was then connected to this webpage so that we easily could store the received answers. The webpage was then posted on NTNU's student webserver, and it's address submitted to our contacts at UNN. They distributed this address to those in their staff that have key card ordering privileges, in other words the future users of the system, and we gave them a deadline of one week the complete the questionnaire. This deadline was later extended by a few days in the interest of the quality of the data collected, as we kept receiving answers after the original deadline was reached.

With a few exceptions it seemed apparent, judging from the received data, that completing the questionnaire via the Internet did not pose too much of a challenge for the users.

[2]PHP is a recursive acronym that is short for PHP:Hypertext Preprocessor. PHP is a highly popular server-side scripting language, which is primarily used for adding advanced functionality to HTML documents, see http://www.php.net

[3]A popular swedish implementation of the SQL-database standard. For more info see http://www.mysql.com

Hopefully, few users abstained from submitting answers because of it's digital format.[4] We have in any case received a significant amount of data to help us draw conclusions. In addition to answering the "select an answer"-type questions, the users also wrote many comments which add quality to the research, and can be useful in the evaluation process.

In the following sections we will present the participants and the important collected data.

4.2.1 Data set reduction

When the deadline had passed, the database contained 85 answers, of these 36 were very incomplete and/or redundant, leaving 49 viable answers. The reason for these incomplete answers is unknown, although we speculate that the users in question may have used the return key instead of the tab key to navigate the fields in the survey, thus submitting an incomplete answer to the database instead of highlighting the next field for input. This seems to have happened repeatedly followed by a complete answer matching the previous incomplete ones, making the latter redundant. We have therefore eliminated these answers and concentrated on analyzing the 49 complete ones.

Figure 4.1 is a cake diagram showing the dispersion of people in each user group, as the term is defined by the questionnaire's question e4(A.6).

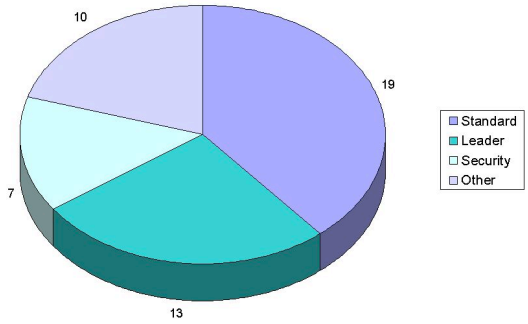

Figure 4.1: The different user groups

4.2.2 Which quality attributes are important?

As described in the beginning of this chapter the complete answers and statistics of the questionnaire is located in the appendix A. This section concerns questions E1 through E21 in the questionnaire. Here we are interested in the user types as defined by the KAS, because it is capable of providing different user interfaces across those user type categories. Some adjustments had to be made to the data to facilitate this, mainly the "Other" category (question S4 in the questionnaire) had to be eliminated. The reason for including this alternative in the questionnaire was to ensure that if the users did not understand our user type categorization they could provide their own definition so that we could choose

[4]This conclusion is only a guess, but an educated one considering the number of possible answers and the number we received

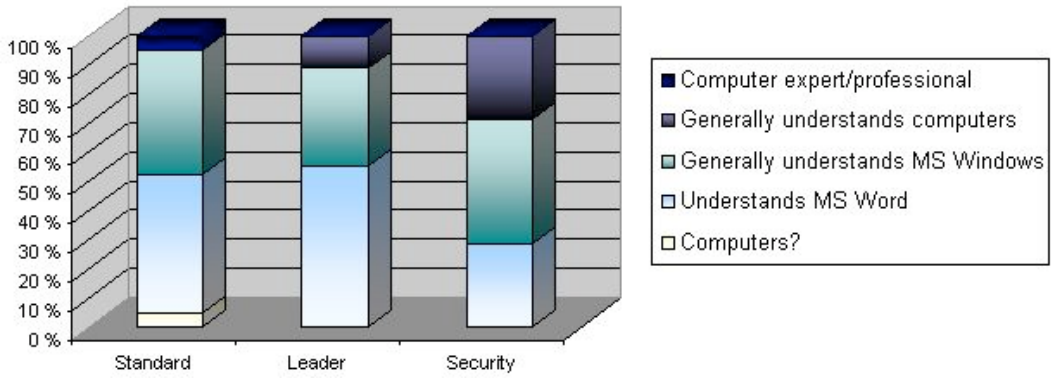

Figure 4.2: The computer skills of the participants

for them accordingly. This seemed to work well and of the 10 replies in the "other" category, 2 were changed to "Standard", 5 to "Leader" and 3 remained unknown. The latter have been removed from all statistical analysis that concern user type categorization, but remain in those that do not. To analyze the answers we have used the mode measure, which is a measure of central tendency according to Wohlin[7], to calculate which quality attributes are most important. The mode represents the most commonly occurring sample and is calculated by counting the number of samples for each value arranging the samples according to this count. We have then calculated relative frequencies and based most charts on this result. We have numbered all the properties the users had to classify, with E#, where E stands for the norwegian word "Egenskaper" (properties) and # is the number of the property. A translated list of these properties can be seen here:

- E1 - Fast response
- E2 - Stability
- E3 - Choice of colors
- E4 - Appearance
- E5 - The presentation of information in a well arranged way
- E6 - Able to use without previous knowledge
- E7 - Error messages that are easy to understand
- E8 - Self explaining tasks
- E9 - Difficult to make mistakes
- E10 - Protection against hackers
- E11 - Protection of personal information
- E12 - Enough information to track possible abuse of the system
- E13 - Changing password often
- E14 - No login
- E15 - User manual in paper form

- E16 - Search after users and employees
- E17 - Show different kind of statistics
- E18 - To see the order status
- E19 - Pre-filled fields
- E20 - Have the option to regret choices
- E21 - Represents qualitative data which will be ignored in the statistical analysis

Each of these properties is connected to one quality attribute as follows:
Performance: E1
Availability: E2
Security: E10, E11, E12, E13, E14
Usability: E3, E4, E5, E6, E7, E8, E9, E15, E18, E19, E20
E16 and E17 are functional requirements that are not so essential in this context.

Usability has most properties connected to it, because this is the intuitively the most important quality attribute for end-users. Modifiability for instance, would not be an important aspect in this context, and therefore it is not represented.

The users were asked to rate all these properties as "unimportant", "less important", "very important" or "indispensable". Figure 4.3 shows the total of all answers. It shows how all the users rated the properties by the four categories, by percentage. The chart is based on the statistics in the tables A.1 to A.20 in appendix A. It became apparent, however, that all user groups has largely the same classification of what is important, as shown by figures 4.4 to 4.9. The charts show the three user categories "Standard", "Leader" and "Security"'s opinions on the importance of the usability properties, followed by the users opinions on the remaining quality properties[5]. The "Security" group's opinions were the only ones that somewhat deviated from the mean, but the number of security users was only 7 out of the total of 49 and therefore it may not be a good representation. But it is also possible that this is because the security personnel has significantly higher computer skills than the other groups. Indeed, computer skill seems to be the only categorization where any sizeable difference in opinion can be observed. Neither age, user type, sex or department seems to notably affect the users preferences. The subjects' computer skill can be seen in figure 4.2.

[5]The four categories of importance have been weighted respectively -1.5, -1, 1 and 1.5. This scale is only created by subjective intuition and the users were unaware of it when they submitted their replies. Therefore these charts can at most be considered a guideline.

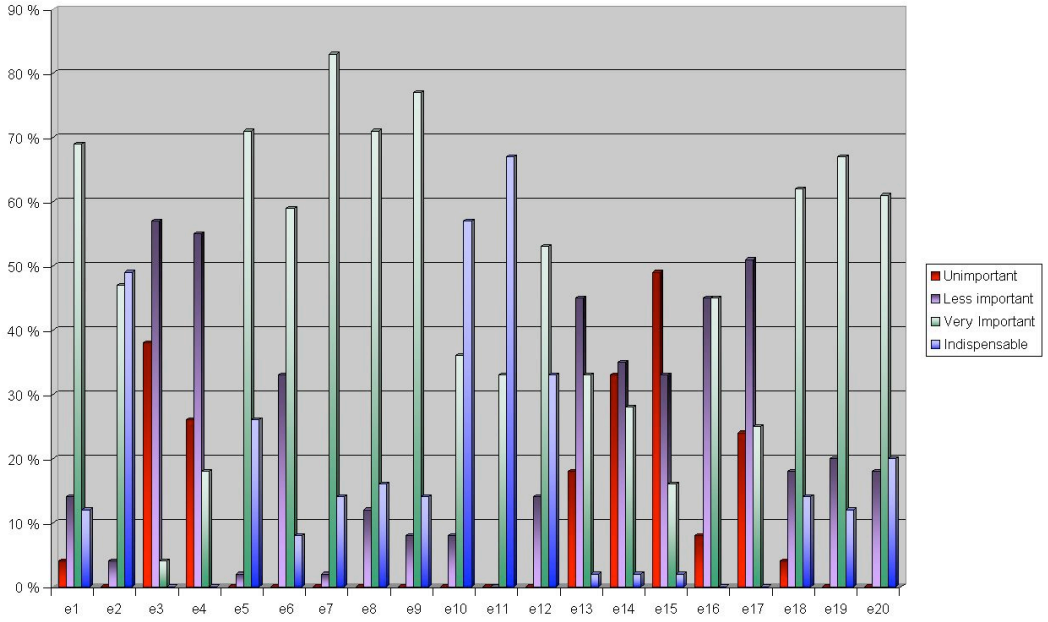

Figure 4.3: The percentage of each attributes importance for all users

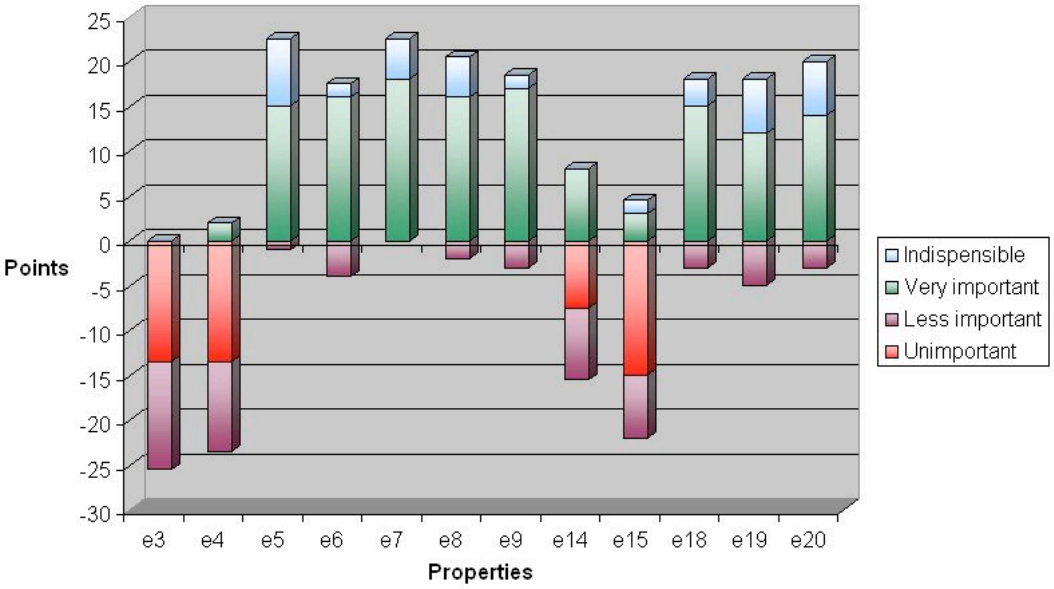

Figure 4.4: Importance of usability attributes according to "Standard" users

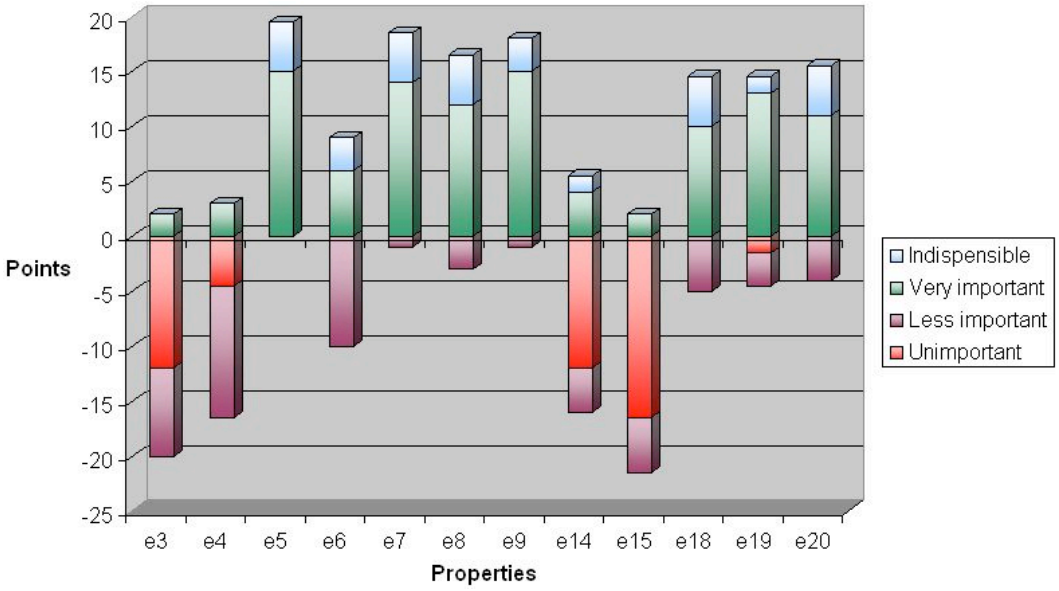

Figure 4.5: Importance of usability attributes according to "Leader" users

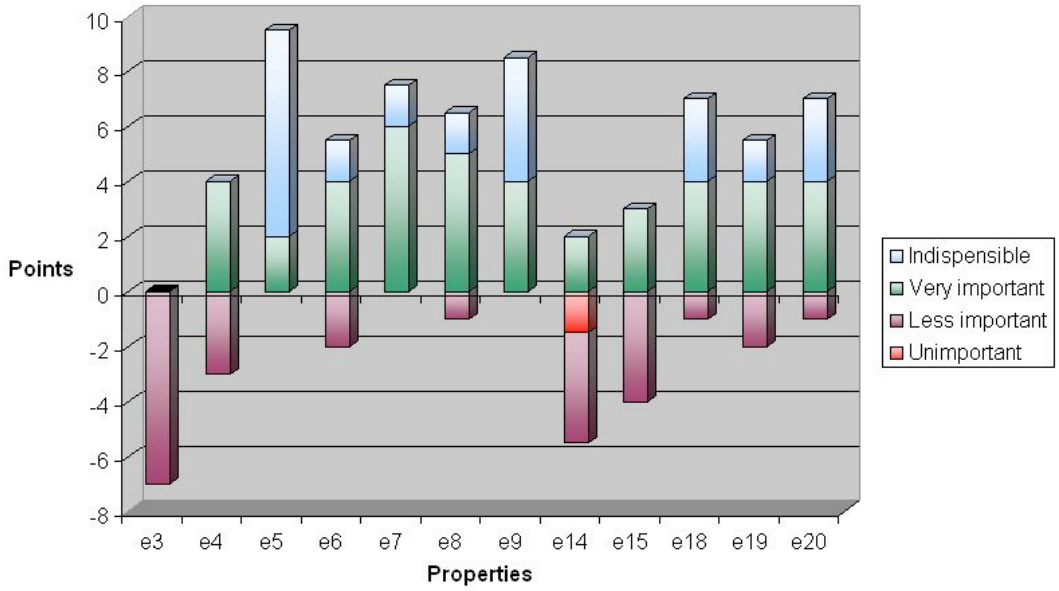

Figure 4.6: Importance of usability attributes according to "Security" users

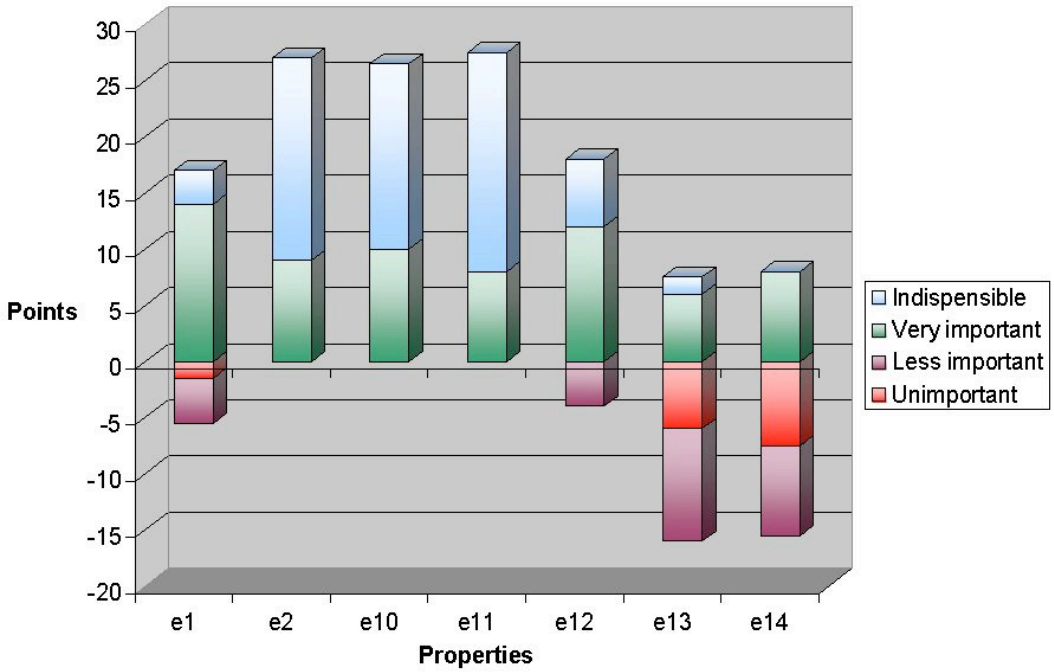

Figure 4.7: Importance of other attributes according to "Standard" users

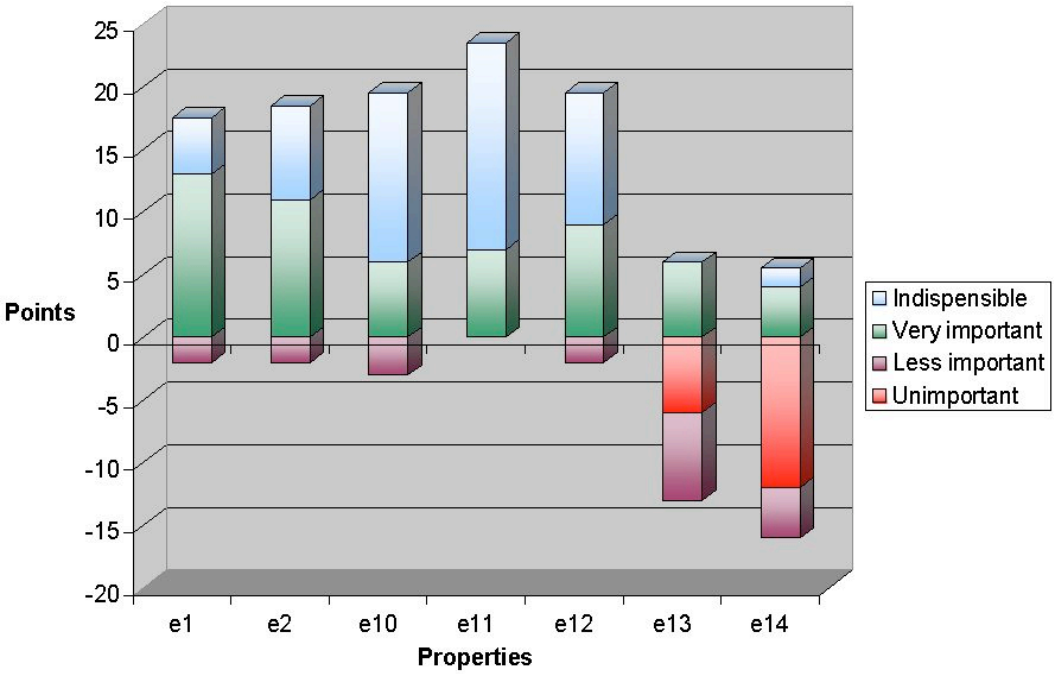

Figure 4.8: Importance of other attributes according to "Leader" users

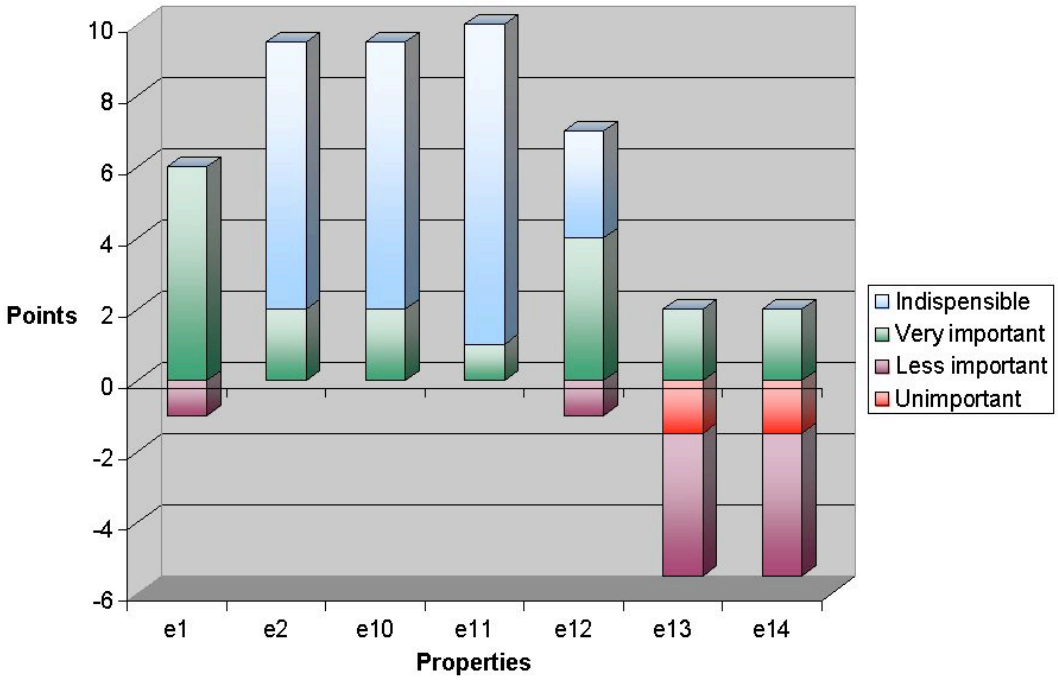

Figure 4.9: Importance of other attributes according to "Security" users

Based on the figures 4.4 to 4.9 and the selected weighted scale, the user groups rated the usability properties' importance according to the following table (in descending order of importance):

Standard	Leader	Security
e11	e11	e11
e2	e5	e10
e10	e12	e5
e7	e7	e2
e5	e9	e9
e8	e10	e7
e20	e2	e12
e9	e1	e18
e18	e8	e20
e12	e20	e8
e6	e19	e1
e19	e18	e6
e1	e6	e19
e14	e13	e4
e13	e14	e15
e15	e4	e14
e4	e3	e13
e3	e15	e3

4.2.3 Quality attribute definitions

Here we were looking for differences in how the different types of users generally define important quality attributes. This section concerns questions b1 through t5 in the questionnaire A. Although all users have answered this section, the questions may not have been clear enough judging from the comments the users provided. At least some users have clearly interpreted this section as "what kind of usability features do you want from the system we are creating" instead of "how do you define usability".

Some of the listed questions are contradictory to each other, and we did this on purpose to easily see if there was contradictory opinions between people.

In all four charts detailing definitions by skill level (in figures 4.11, 4.12, 4.13 and 4.14), the three skill levels correspond to those shown in figure 4.2, but since only one person were represented in both the categories "Computers?" and "Expert", these two categories have been removed. The person who defined his/her skill as the former has been added to the "Can create documents category" and the latter has been added to the "General understanding of computers" category. Thus the categories in the charts can be defined as follows:

1. Low skill, can at most create text documents
2. Medium skill, generally understands MS Windows
3. High skill, generally understands computers or is an expert

Usability

The figures 4.10 and 4.11 details how the users defined usability ordered by the type of user, and computer skills, respectively. The numbers on the x-axis correspond to the following:

1. Looks like other computer systems
2. Has nice colors
3. Has well arranged menus
4. Has an abundance of help functions and Wizards
5. Does not have many wizards and instead gets directly to the point
6. Has a user manual on paper
7. Does not require a user manual
8. Hides prospective errors from the user
9. Shows all errors to the user

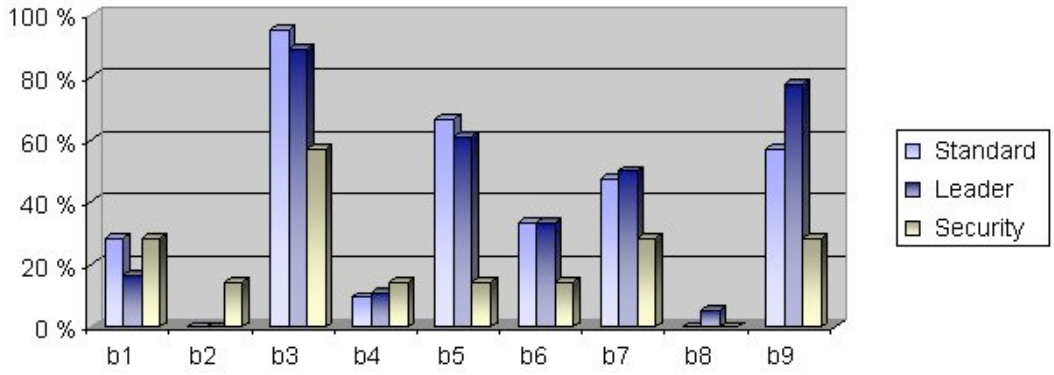

Figure 4.10: Users' opinions of usability ordered by user group

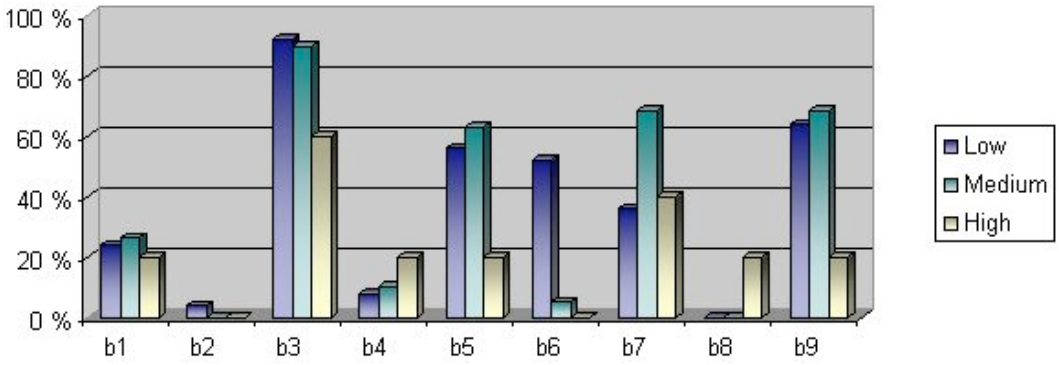

Figure 4.11: Users' opinions of usability ordered by skill level

Performance

The charts in figure 4.12 details how the users defined performance ordered by the type of user, and computer skills, respectively. The numbers on the x-axis correspond to the following:

1. The time it takes from you click on an element till the computer finishes working
2. How fast a program is opened
3. The number of programs you can keep open simultaneously

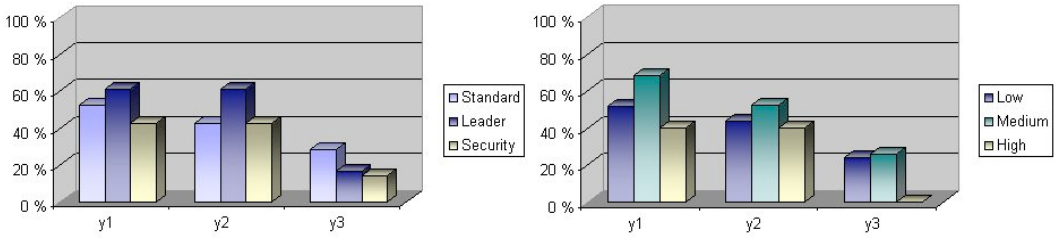

Figure 4.12: Users opinions of performance attributes by user type and computer skill

Security

The charts in figure 4.13 details how the users defined security ordered by the type of user, and computer skills, respectively. The numbers on the x-axis correspond to the following:

1. That it checks in some way that a person is who he/she claims to be
2. That a user is unable to access more information than necessary
3. That it is protected against hackers
4. That is protected against viruses etc.
5. That personal information is kept confidential

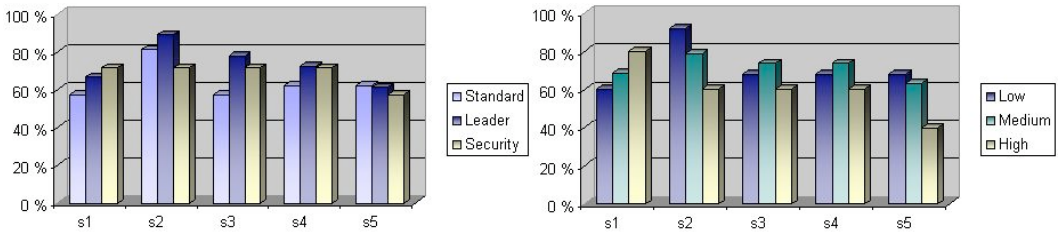

Figure 4.13: Users opinions of security attributes by user type and computer skill

Availability

The charts in figure 4.14 details how the users defined availability ordered by the type of user, and computer skills, respectively. The numbers on the x-axis correspond to the following:

1. That information can be accessed in several languages
2. That it is accessible from multiple locations, ie. from your office, at home etc.
3. That it detects errors when they occur, and is able to correct them
4. That it is not out of commission

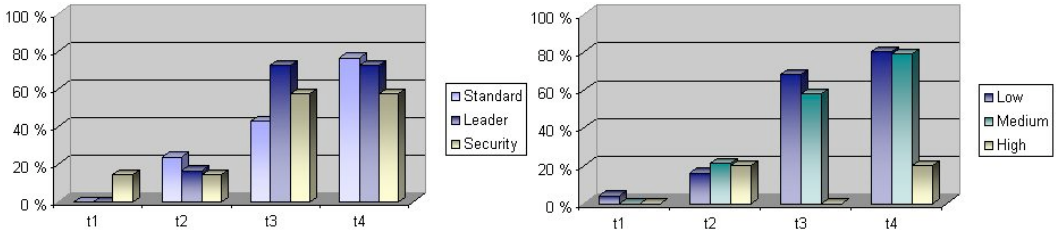

Figure 4.14: Users opinions of availability attributes by user type and computer skill

4.2.4 Summary

We will now summarize some notable results from the questionnaire.

Which quality attributes are important?

Usability

Security personnel thought that it was extremely important that it was difficult to make mistakes in the system. They also generally found all usability properties significantly more important than the other two groups. Appearance was relatively a lot more important to security personell, while it was considered one of the least important by the other users. This, Choice of colors and user manuals scored the lowest on average. Department leaders couldn't care less about user manuals, but security viewed them as quite important.

All users viewed e5, e7, e8, e9, e18, e19 and e20 as the most important on average, with a few exceptions they considered them very important or indispensable.

Security

Also here security personnel viewed the properties are more important on average. An exception is that system contains enough information to track abuse, which the leaders found more important the both other groups. Users seemed to disagree amongst their own group about the importance of changing passwords often and not being required to log in, but most found them of very low importance.

Almost all users viewed e10, e11 and e12 to be of extremely high importance. All but one of the security personnel found e11 to be indispensable.

Performance, Availability

There was some disagreement internally in the groups on the importance of fast response. As a group, leaders were most positive and standards least positive, but in general most users viewed it as very important. All but two of security found stability to be indispensable, the other groups were a little less optimistic, but in general this was considered very important or indispensable.

Quality attribute definitions

Usability

b2, b4 and b8 are generally not defined as usability. Most security personnel do not define any of the porperties as usability, the only property that scored over 50% was well arranged menus which scored almost 60%. This scored highest amongst all users, about 90% by the other two groups. There are striking similarities between ordering by user groups and ordering by computer skill, indeed security personnel have higher computer skills than the other groups. In general the higher the skill, the lower the percentage have defined properties as usability. Over 50% of those with low skill define user manual on paper as usability, but almost none of the others do.

On average, b3, b5 and b9 is defined as usability, b6 and b7 slightly below 50%, while the rest is on average not defined as usability.

Performance

There is also here striking similarities between user group and computer skill sorting, although the lack of difference in opinion between all subjects can be the cause for this. About half of the users defined response time and the time it takes to open a progran as performance.

Security

Again, sorting by user group or skill seems to have little effect. There is a notably larger percentage who define these attributes as security than in the other categories, all security attributes in general score from 60-80%. Authentication(s1) seems to be increasingly defined as security as skill level rises, while the opposite can be observed of authorization(s2) and confidentiality(s5). The high computer skill group only consists of 5 subjects, so this can be coincidental.

Availability

While the users largely agree sorted as user groups, there are some differences when sorted by skill. While those with low and medium skill mostly define error detection/correction and that the system is not out of commission as availability, those will high skill do not.

In general t3 and t4 is defined as availability.

Part III

Software testing

Chapter 5

Quality attribute analysis

Here we will conduct an ATAM analysis based on the important attributes from chapter 4.1 and 4.2. The analysis has been scaled down from the one presented in [2] to a format better suited for this project, for instance the KAS does not have a documented software architecture. It has also not been conducted in the presence of all stakeholders for practical reasons, but their interests are represented through the data from the empirical studies. It primarily includes an analysis of the trade-offs and risks between the most important quality attributes that have been identified throughout the empirical studies, and it's purpose in this report is to document the viability of the already completed and future additions to the KAS by analyzing quality attributes and asses the trade-offs between them. It can also be seen as a test of high-level design errors, which tend to relatively increase with respect to low-level coding errors as the system grows in size [5]. The ATAM is in our experience good at rooting out possible high-level architecture type errors. Low-level internal testing will be done in the following chapters.

5.1 Introduction

The important stakeholders and their agendas in this development project are:

- The development team (The authors of this report)
  Goal: To complete the project in cooperation with both UNN and NTNU.
- The contacts/administrators (Svendsen and Lyngmo, UNN)
  Goal: To get the system up and running to enable a more effective and controllable operation at the hospital.
- The university (NTNU)
  Goal: To get its students to conduct some research in the area of software development.
- The end users (Employees at UNN)
  Goal: To get the system up and running for a more streamlined access card ordering system.

5.2 Attribute utility tree

The utility tree in figure 5.1 presents some important scenarios that may come to pass in the system. The scenarios are made from the most important quality attributes defined

by the users in the surveys. The numbers on each scenario represent probability of happening, importance and difficulty in implementing respectively. 1 is lowest, 6 is highest. For example, (6,3,1) means the scenario will definitely occur and probably often(6), it is of medium importance that the system can handle it(3), and it is very easy to facilitate(1).

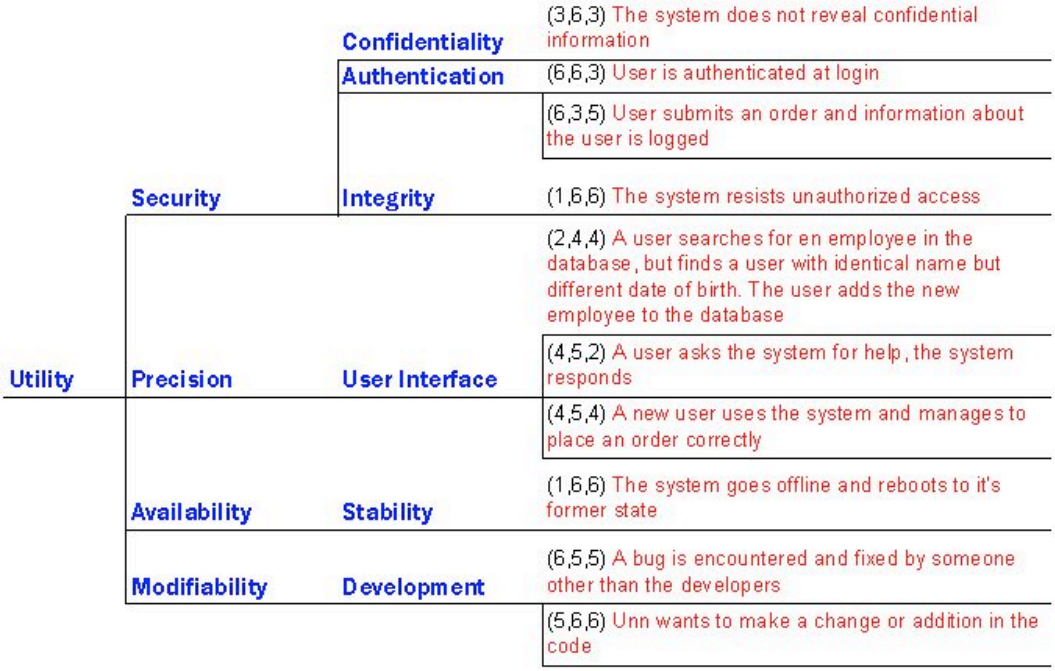
Utility
Security
Confidentiality
(3,6,3) The system does not reveal confidential information
Authentication
(6,6,3) User is authenticated at login
(6,3,5) User submits an order and information about the user is logged
Integrity
(1,6,6) The system resists unauthorized access
Precision
User Interface
(2,4,4) A user searches for en employee in the database, but finds a user with identical name but different date of birth. The user adds the new employee to the database
(4,5,2) A user asks the system for help, the system responds
(4,5,4) A new user uses the system and manages to place an order correctly
Availability
Stability
(1,6,6) The system goes offline and reboots to it's former state
Modifiability
Development
(6,5,5) A bug is encountered and fixed by someone other than the developers
(5,6,6) Unn wants to make a change or addition in the code

Figure 5.1: Utility tree with important scenarios

5.3 Analysis of architectural approach

By considering the scenarios' ranking in utility tree, we will here look at the most important ones in detail. The scenarios "The system resists unauthorized access" and "The system goes offline and reboots to its previous state", while very important, are largely dependent on other factors outside of the development team's control, and will therefore not be considered here. The security issues of the developers concern are mainly those associated with user authentication, while the other two fall under the hospital's IT-department.
Security, usability and modifiability are, as stated by the users in the surveys, the most important quality attributes, and we will analyze scenarios based on these three quality attributes to find out if there are any trade-offs between them.

5.3.1 "UNN wants to make a change or addition in the code"

Scenario:	Unn wants to make a change or addition in the code			
Attribute(s):	Modifiability			
Environment:	Normal operation			
Stimulus:	Change in functionality is requested			
Response:	The code is changed without altering the rest of the system			
Archtectural decision	**Sensitivity**	**Tradeoff**	**Risk**	**Nonrisk**
Using configuration files	S1			N1
Hide information	S2			N2
Restrict communication paths	S3			N3
Abstract common services	S4	T1		N4
Reasoning:	It will, according to the employers, at some point be necessary to change the code, most probably for the purpose of adding functionality			

Figure 5.2: Scenario 1

5.3.2 "User is authenticated at login"

Scenario:	User is authenticated at login			
Attribute(s):	Security			
Environment:	Normal operation			
Stimulus:	User tries to access system services			
Response:	The user is authenticated and given access to the applicable resources			
Archtectural decision	**Sensitivity**	**Tradeoff**	**Risk**	**Nonrisk**
User authentication	S5	T2	R1	
User authorization	S6	T3	R2	
Reasoning:	Only authorized users are meant to use the system, and logging in is thus the first task a user must complete.			

Figure 5.3: Scenario 2

5.3.3 "A new user uses the system and manages to place and order correctly"

Scenario:	A new user uses the system and manages to place and order correctly			
Attribute(s):	Precision			
Environment:	Normal operation			
Stimulus:	User wants to use the system efficiently			
Response:	System tries to support its efficient use			
Archtectural decision	**Sensitivity**	**Tradeoff**	**Risk**	**Nonrisk**
Re-use of information	S7		R3	
Intuitive placement and naming of buttons and information	S8			N5
Input restrictions on data fields	S9			N6
Reasoning:	This will happen frequently in the early stages in the systems operation, and at some lower rate continously after that			

Figure 5.4: Scenario 3

5.4 Sensitivity points

Figure 5.5 describes which quality attributes are significantly affected by the selected architectural decisions, and elaborates these decisions to a certain degree.

S1	Positively affects modifiability because it localizes all login information, and security by not having passwords or server information scattered in the code.
S2	Has positive effect on modifiability by limiting the amount of data available outside each module.
S3	Similar to S2, by restricting the number of modules depending on eachother the number of possible ripple effects is reduced thus supporting modifiability. It also makes module testing easier.
S4	By abstracting commonly used services as functions, changes are localized and code size is reduced. Very positive for modifiability and testability
S5	Makes the system much safer by requiring the user to identify him- or herself. Enables the use of digital "fingerprints" or signatures on any action the user takes in the system. This is essential for a secure system.
S6	Positively affects security by limiting a user's access to only the resources he or she needs
S7	By re-using any information the system already has or the user has manually entered earlier, the risk of human error in user tasks is reduced. This positively affects precision, and security because it helps ensure that users do not enter false information
S8	Makes the system easier to use and the interface more intuitive and self-explanatory, thus increasing precision
S9	Positively affects precision by ensuring that the user enters correct information, and thus making life easier for the user, and limits the amount of errors the system has to handle effectively

Figure 5.5: Sensitivity points

5.5 Trade-off points

Sensitivity points that have trade-off effects on some quality attributes are presented and explained in figure 5.6.

T1	Many levels of abstraction adversely affects performance
T2	A trade-off is that the user has to remember a username/password, which he or she may be required to change at given intervals, and the user has to traverse the obstacle of logging in to the system prior to using it. Thus negatively affecting usability.
T3	Keeping track of several user classes and their access privileges severely complicates the code, thus negatively affecting testability and modifiability

Figure 5.6: Trade-off points

5.6 Risks and non-risks

Risks are architectural decision that can have a negative effect on quality attributes and the system as a whole. Non-risks are decisions that do not pose any threat to the system's quality. These can be seen in figure 5.7

R1	Too strict password conventions will probably lead to users finding their own ways of making the system more practical, leading to extreme measures like posting usable passwords on bulletin boards. This would severely threaten security, which is a primary concern
R2	The fairly complex restrictions required on access to the different parts of the system requires extensive knowledge of the system in the event of a change or an addition to avoid compromising security. This negatively affects modifiability.
R3	Re-using information requires keeping track of information which increases the amount of data that is visible and changeable outside modules, thus negatively affecting modifiability and testability. It also increases the amount of cross-module dependencies, increasing the chance of ripple effects, further compromising modifiability
N1	No risk because of no significant negative effects
N2	No risk because of no significant negative effects
N3	No risk because of no significant negative effects
N4	No risk because performance is not an important factor
N5	No risk because of no significant negative effects
N6	No risk because of no significant negative effects

Figure 5.7: Risks and non-risks

5.7 Risk themes

It is clear that notably the security requirements and the usability requirements, and the architectural decisions and tactics used to fulfill them, have adverse effects on the systems modifiability. Several also have a negative effect on performance, notably response time.

5.8 Conclusion

Security and usability are the primary concerns, and although modifiability is important, the former must take priority. However care must be taken to adversely affect modifiability as little as possible. Performance is not a priority issue, and considering the relatively low amount of users, it should not become a problem anyway. The hardware and operating environment for the KAS is already scaled for handling much more time-critical and performance-hungry systems. The normally very high security considerations normally associated with web-based systems are very much laxed by keeping the entire system disconnected with the internet, it will only be accessible through a reasonably secure intranet.

Chapter 6

Test planning

In the testing process we have decided to use the Goal/Question/Metric method as defined in 3.3.1. The method provides a systematic approach to the process, which we believe will be very helpful considering the lack of well-defined methods for software testing in general [5]. The goal of this section is internal testing of the system, using human code inspection and user testing as far as it can be done from the developers point of view, with the purpose of uncovering low-level coding errors and omissions. We will also try to asses the quality of the user interface. We are aware that code testing should be done by other parties, for example [5] warns about developers testing their own code. However, considering the scope of the project it is not an option to include an independent test group. [5] suggests error testing should be aimed at uncovering errors to be successful, as opposed to trying to document that the system is error-free. Several of the following metrics should also be tested by end-users to be effective, this is planned for future work.

6.1 Goal

We have formulated a research goal based on the results of the empirical studies, on the form suggested by Wohlin [7]. The goal is broad, but reasonable, and it provides a clear direction for further study. We define it as follows:

Analyze the *Keycard Administration System*
for the purpose of *testing if it fulfills the requested quality requirements*
with respect to *specifically usability, security and modifiability*
as seen from *the software developers' point of view*
in the context of *an analysis project at NTNU*

By requested quality requirements we mean those which are defined in the original requirement specification and the ones conveyed by the different types of users through the survey and interview.

6.2 Questions

To reach our goal we must define questions that, when answered, brings us closer to it. We feel that the following questions capture most of the important issues involved in reaching the goal.

Name	M1: KAS precision
Definition	The user should be able to precisely determine how to enter data into the system
Procedure for measuring	Measure if the syntax of the various input data is clearly defined by counting the number of special syntax input fields that have an explanation (A) compared to the number of fields requiring special syntax (B). X=A/B, the closer X is to 1 the more precise.
Expected value	From our own experience this has been a priority during the development phase, we therefore expect X to be close to 1.

Table 6.1: Metric 1, KAS precision

Question 1 *How intuitive is the KAS interface?*

This question covers many aspects concerning GUI[1] design and the front-end the users will need to work with. This question is measured by M1, M2, M3, M4 and M5.

Question 2 *How secure is the KAS?*

Security is a vital issue in the KAS, and is broadly addressed by this question. It is measured by M1, M2, M7 and M8.

Question 3 *How will the KAS respond to changes or additions in the code?*

The system will inevitably require change to keep up with the changing processes it is supposed to handle, and to incorporate planned future changes. The system's modifiability is measured by M6.

6.3 Metrics

The metrics are listed in the following tables as proposed by Stålhane [6]. Since all metrics will be part of an internal testing session, and will be tested simultaneously, they will have the same "When to measure"-field, namely "During the test session". Thus we feel it is unnecessary to include this field in all the tables. The metrics are loosely based on internal metrics suggested in the ISO 9126-3 standard [1].

[1]Graphical User Interface; how the system is visible to the user, and the tools and controls the user must use to communicate with the system

Name	M2: Error tolerance
Definition	The system should be able to handle erroneous user input correctly
Procedure for measuring	This will be a more qualitative test with different test data for the different kinds of data fields.
Expected value	A subjective, qualitative judgement will have to be made as to how error tolerant the system is. However, error tolerant code is time-consuming to produce and this has not been a priority, so we expect a significant amount of erroneous input to be poorly handled.

Table 6.2: Metric 2, Error tolerance

Name	M3: Wizards
Definition	A software system can rely heavily on wizards for user tasks/processes, or it can proceed more directly to the point. We define a wizard as a step-by-step guide that verbosely describes each step in the process.
Procedure for measuring	Count the number of wizards in the ordering system (A) and compare to the total number of user processes in the ordering system (B). X=A/B, the closer X is to 1 the more intuitive it is to user who like wizards.
Expected value	The priority has been to make the system intuitive without the need for wizards, thus the expected result is X = 0

Table 6.3: Metric 3, Wizards

Name	M4: Error messages
Definition	The error messages of the program should be meaningful, non-abusive and devoid of computer gibberish.
Procedure for measuring	Count the number of error messages that are possible to understand without any special computer knowledge (A), compared to the total number of error messages (B). X=A/B, the closer X is to 1 the more meaningful are the error messages.
Expected value	Error messages have been a priority, but the system has low exception handling capability, so we expect X to be approximately 0.7

Table 6.4: Metric 4, Error messages

Name	M5: Feedback
Definition	When a user performs an action, the system should respond in an intuitive way. For example, if you push a register button, a clear confirmation message or action should be presented. This is important in order to avoid the user misunderstanding and repeating an action needlessly, which could also lead to database errors.
Procedure for measuring	Count the number of confirm/register functions that produce a clear confirmation message or action (A) and compare to the total number of confirm/register functions (B). X=A/B, the closer X is to 1 the better the feedback of the system to the user.
Expected value	This has been a priority and we expect X to be at least 0.9

Table 6.5: Metric 5, Feedback

Name	M6: Modification localization
Definition	The software system will eventually have to be altered in some way, be it additions, changes or bug fixing. To minimize this effort, a change should have minimal ripple effects in the system.
Procedure for measuring	Count the number of modules/webpages any global variable (except the one that keeps track of the logged in user, because it affects all pages and logically has to be considered in any modification) affects (A) and compare to the total number of modules/webpages (B). X=1-A/B, the closer X is to 1 the less the impact of modification.
Expected value	Hard to determine by experience as modifiability has not been a high priority, but we expect X to be approximately 0.75

Table 6.6: Metric 6, Modification localization

Name	M7: Authorization
Definition	Users should only have access to required information, and any other information should be hidden.
Procedure for measuring	Count how many processes each user type can access (A) and compare to the number they should have access to (B). X=A/B, if X is less than 1 security has not been compromised but has lacking functionality, if X is larger than 1 security has been compromised.
Expected value	We expect this value to be 1

Table 6.7: Metric 7, Authorization

Name	M8: Authentication
Definition	If the system can not authenticate you as a user, you should not have access to it.
Procedure for measuring	Count how many pages that can be accessed without being authenticated (A) and compare to the total number of pages requiring authentication (B). X=1-A/B, the closer X is to 1 the more secure the system is.
Expected value	This has been a high priority, we expect this number to be 1

Table 6.8: Metric 8, Authentication

Chapter 7

Test results

The results from the internal testing phase will be presented in this chapter. It will specifically describe the outcome of applying each metric defined in the previous chapter, and how it affects the questions and ultimately the goal.

7.1 Metrics

The results from each metric is presented here

7.1.1 M1: KAS precision

A = 8
B = 11
X = 0,73
Expected value: Close to 1

Two of the three fields that lacked clearly defined syntax descriptions could be considered self-explanatory, since an identical field with a complete description was located nearby. If these were considered sufficiently described, X would be 0,91.

Conclusion: The user will in most cases be able to determine how to enter data into the system.

7.1.2 M2: Error tolerance

Text fields are fields that accept large amounts of text and is used for anything from names to department descriptions. This is some of the test data for the text type fields:

In-data	Reaction
123 asd	works
123-asd	works
asd_123	works
asdś	unhandled exception
asd123	works
asdÏ23¨	works
asd123$\tilde{2}$3	works

The text fields surprisingly accepted all input data, erroneous or not, with the exception of the apostrophe character which returns an unhandled exception. This in practice

means that the user will not see any error messages for any erroneous input, except for the apostrophe character, which should not be considered an illegal character thus leading to unnecessary hassle for the user. There are, for example, people whose names contain this character. We asses this test result as mediocre.

Number fields are used for Key card numbers. This is the test data for the number type fields:

In-data	Reaction
-1,999	works
-1.999	unhandled exception
-1	works
0	works
1	works
1,999	works
2147483647	works
2147483648	unhandled exception
a	unhandled exception

Negative numbers and decimal numbers are not used and could have been refused by the system, but this is not a problem. a and -1.999 are not numbers as far as Microsoft is concerned and considered erroneous (should be a comma in the latter). Not problematic, but will return an unhandled exception. 2$\hat{3}$1 is also a Microsoft limitation to number size. Not a problem, but a larger number will return an unhandled exception. The number fields will only be used by security or admin personnel which should not have any trouble with these issues or any interest in trying to create problems in the system. We asses this test result as good.

There are 6 date-fields and 5 datetime-fields in the system, with their obvious uses. They were all tested, but all gave the same result, therefore we present only the interesting results from the datetime fields. Test data:

In-data	Definition	Reaction
2005-12-31 23:59	last timestamp this year	works
2006-01-01 24:88	non-existing timestamp	unhandled exception
2005-13-01 12:00	non-existing month	unhandled exception
14.12.2005 12:00	todays date, wrong format	handled by error message
123 asd	wrong format	handled by error message
no data		handled by error message

These are the fields we suspect will be most susceptible to errors, considering their awkward indata syntax requirements and the fact that all users will use them frequently. Therefore an error-handling scheme has been created, that captures all data in the wrong format. It does not, however, capture data that has the right format but wrong content, i.e. a non-existing time of day, which will return an unhandled exception. This result is as expected and considered fairly good, but improvements are planned.

Conclusion: We feel that the system has medium error tolerance, and can produce complex error messages that are unhelpful to the users. There is also a chance that erroneous data is accepted in some cases.

7.1.3 M3: Wizards

A = 0
B = 30
X = 0
Expected value: 0

Conclusion: The KAS has a unique, specially designed user interface which gives the user all the information he/she needs but nothing more (an assessment based on the intuition and experience of the developers and the administrators, thus not necessarily true, but thought through). It has been made a point that the system should not need wizards for the users to complete their tasks.

7.1.4 M4: Error messages

A = 18
B = 22
X = 0,818181818
Expected value: Approx. 0,7

Many of the error messages occurred multiple places in the code, only one of each has been taken into consideration.

Conclusion: The error messages in the system are very informative and meaningful, except for the unhandled exceptions.

7.1.5 M5: Feedback

A = 31
B = 31
X = 1
Expected value: At least 0,9

Two of these confirmation messages were clear, but did not live up to the standard of the rest.

Conclusion: The system responds to important user actions in very intuitive and consistent ways.

7.1.6 M6: Modification localization

A = 24
B = 31
X = 0,225806452
Expected value: Approx. 0,75

Five global variables that are used for user authentication and info/error messages were not included as they are considered essential to the system and will have to be considered in the event of a code modification.

Conclusion: The system is surprisingly prone to problems/errors from ripple effects in the event of code modification.

7.1.7 M7: Authorization

For all five user groups, the result was the same, X = 1 as expected, with the exception of the result from metric M8, which allow access from anyone.

Conclusion: The user groups have correctly authorized access to the system's resources

7.1.8 M8: Authentication

A = 3
B = 30
X = 0,9
Expected value: 1

Three webpages allowed unauthorized access, of which two are considered a danger to security.

Conclusion: The system is highly secure with respect to authentication, but it has a few security breaches.

7.2 Questions

We will now summarize the results from the application of the metrics to try to answer the questions.

7.2.1 Question 1

The metrics M1, M2, M3, M4 and M5 all point to that the KAS interface is very intuitive, with some exceptions, notably:

1. The date and datetime fields that have very awkward syntax, which is not user friendly.
2. Syntax errors here are handled, but incorrect data in correct syntax is not and will lead to unhelpful error messages.
3. Erroneous input in the other types of fields are not handled at all and will lead to unhelpful error messages or errors in the database.

7.2.2 Question 2

The metrics M1, M2, M7 and M8 points to that the system is reasonably secure, but has some issues that need to be addressed, notably:

1. Some errors can be input into the database and can potentially harm the system in unpredictable ways
2. There are a few security breaches that can be accessed by unauthenticated users

7.2.3 Question 3

The metric M6 indicates that the system may not respond well to changes or additions in the code, notably because there are many global properties that can be changed from anywhere in the system which can cause a number of problems.

7.3 Goal

As seen from the developers' point of view in the context of this analysis project we conclude that the system with a few modifications will have high quality with respect to security and usability, but more work most be done to achieve a satisfactory level of modifiability.

Part IV

Final results

Chapter 8

Evaluation and discussion

In this chapter we evaluate and discuss the results from the previous chapters. It starts off with a discussion of validity of the methods used and results received throughout the project. It is especially important to validate qualitative data to ensure the correct conclusions. Furthermore, this chapter evaluates the important results obtained during this project.

8.1 Validation

A discussion of validity of both the methods used, the decisions made and the results, is fundamental to be sure that the data is reasonable and trustworthy. We must ensure that the data has been collected correctly in a way so that we can draw the right conclusions. There exists many threats that can affect different types of projects, be it case studies, surveys or experiments, both positively and negatively. Wohlin [7] discusses validity threats especially for experiments, but some of these might also be a threat to other empirical strategies as well, for example surveys as in our case. Wohlins list is by far not complete and not all of the threats in the list are applicable to all experiments either. These threats are divided into four groups, namely conclusion, internal, construction and external validity. We will not use this categorization directly and nor will we describe each threat in detail, as they can be seen in Wohlins work, however we will use his list as a basis to find the validity threats in our project. This validation mainly concerns the results of the questionnaire, and the decisions made during the statistical analysis.

Considering the total number of replies to the questionnaire, 49, it seems like we have enough answers to draw some overall conclusions. We didn't expect this many answers in the first place, but the more the better and this amount should provide a solid fundament to back up our conclusions. The amount of replies also suggest that few if any people have abstained from completing the questionnaire, thus the decision to distribute it in digital format over the Internet seems to have been a good one.

When we divide the replies received between the different user groups, we can however point out a threat. This threat concerns the low number of security personnel. We only received 7 replies from security, although this is not necessarily a low participation ratio as they are fewer in number than the other user groups. It still makes it harder to do any meaningful statistical analysis, and the security personnel has lot of responsibility and is an essential user group in the system. The actual selection of security personnel that completed the questionnaire probably affects any statistically based conclusions to a

large extent.

The decision to include the representatives of the "other" user group from the questionnaire into categories that better relate to the KAS' user groups (since "other" is not a valid user group of the system) was a good opportunity to get a better data foundation in the remaining user groups. This was possible to do because of these users' own definitions of their respective user groups. However, three of these users did not supply their own definitions and we were thus unable to classify them. Their replies are therefore included in overall results, but are removed from any user group based analysis. The reason for allowing the users to provide their own definitions on their user groups was that, by experience, there is always someone that do not classify themselves to the proposed options, in which case we were interested in their own definitions. Second, since it was hard to create choices in the questionnaire that couldn't be misunderstood, we felt it important to capture such misunderstandings by means of a field for one's own alternative. Indeed, more than 20% of the replies were classified as "other", and of these 70% had it's own definition.

Only one person was represented in the "Computers?" and one in the "Computer expert" categories. These were added to the "Understands MS Word" and "Generally understand computers" categories, respectively. Otherwise their opinions would have had too much weight in their own categories, either 0 or 100% depending on their answers.

Wohlin talks about "fishing". "Fishing" describes the phenomenon were researchers look for a specific result or outcome and possibly overlook other results. The forming of the questions in the questionnaire could be a threat to the results, for example the "other" field described above. The questions have been formed by us, the developers of the system, and we have knowledge of how the system works and what is crucially important. Maybe the questions were too centered on specific areas. For example, we were trying to determine which quality attributes were the most important, but usability and security was given a lot more space than the others, and some were not even included. The reasons for this was that end users intuitively do not concern themselves with internal aspects like code portability. By experience they are however very interested in usability, and finally the requirements specification and the wishes of the administrators have been given priority. But we should maybe have made a bigger effort to include other quality attributes.

The questionnaire was, during the whole session, openly available on the Internet. This effectively means anybody who could find it could have answered it, which would have degraded the quality of the total material. We considered the probability of this happening but it seemed unlikely due to the fact that no one from the outside would be looking for it, and should they find it by chance there would be little reason to submit a reply. And after reviewing the data there was no indication of any "false" answers.

More than 30 incomplete answers were received, and almost all only had the first few fields filled out. What happened? It is not easy to say, but we choose to believe the reason described earlier in this report. It seems like a few persons, one in particular, experienced problems with navigating the questionnaire and submitted incomplete answers before submitting a complete one. These answers have been removed as they seemed to be both redundant, and also so incomplete that they would not have made any contribution to the material anyway. "Outliners", as in data that are very different for no apparent rea-

son, do not provide any valid material anyway, and can be safely removed [7].

Another threat to data validity is the probability that the subjects has misunderstood some of the questions. Indeed, based on the comments some subjects provided in their replies, some have misunderstood the questions b1 through t5 and conveyed their wishes about the system(KAS) when they should have been thinking about any system in general. This section in the questionnaire was designed in such a way to capture the users' more general opinions and make it possible to generalize these opinions to a larger population. Such generalization will now be hard to do.

In the figures from 4.7 to 4.11 where ordering by skill is performed, the total population is 46 compared to 49 in all the other figures. This could be a threat, at least it could make the figures difficult to compare since they are based on a different basis. In addition, the figures 4.4 to 4.9 only gives an indication on the importance of the different properties, and the dispersion of the different classifications of importance. These three figures are also based on different numbers of answers, and therefore also may be difficult to compare.

8.2 Evaluation

The research in this project has basically been divided into two major parts, the empirical studies and the testing. The first part concerned two areas of interests, the importance and definition of quality attributes. The goal of the first area was to find the stakeholders most important quality requirements to a system. The goal of the second area of interest was to find the stakeholders definitions of some well-known quality attributes in modern software architecture. The second major part of the problem concerned also two areas of interests, the analysis and testing of the system. The first area of the second part was to find out if the requested attributes supports or contradict each other, and what must be done if they contradict. This should also uncover high-level design errors. The last area concerns low-level testing of the KAS against the most important quality attributes to see if these requirements have been met.

We will now discuss each area in more detail.

8.2.1 Empirical studies

First we discuss issues from the empirical studies.

Interview

The outcome of the interview was pretty much as expected, the KAS quality requirements are largely the same as they were in the original requirements specification. But a clearer picture of the functional requirements mapped to quality attributes has been established, and some new issues have emerged.

The interview subjects were asked how they thought we could test the system in a meaningful way, and they suggested that we could have a simulated user test phase. This could be conducted in a day on site, with a selected group of users that are given preplanned tasks to complete in the system, with or without training. Then a lot of useful feedback would be produced while keeping the cost and risk to a minimum. However, we subsequently decided to keep our testing efforts in this project to the internal level,

and postpone user testing to further studies. Mainly because of time constraints, but also to identify as many errors as possible before subjecting the system to end users.

It became clear that extensive technical documentation would have to be created to enable any other parties to further modify/extend the system at a later stage. UNN does not want to be locked to the current developers, and this is the basis for the concerns of modifiability issues in the KAS. This has not previously been a priority. Some form of user aids would also be preferable.

Of the new issues that have emerged, one can at least be worth mentioning. Another new computer system has been introduced at UNN, containing employee information, making, including the KAS, a total of at least three databases containing such information. This will inevitably become a database-updating nightmare if it is not taken seriously. Preferably only one of these databases should be kept updated, while the others received data from the updated database.

All new functional requirements aside, it has been made clear that the top priority is getting the KAS operational. This prioritization will, at least, have a significant impact on any further work on the KAS. No functional requirement shall be allowed to hinder the system in becoming operational.

Questionnaire

The Questionnaire generated a large amount of data, and much work was required to evaluate and present the statistics.

The users were largely in agreement of which properties in the system they thought was important and which was not. A few exceptions are e14 and e15. Some leaders viewed it as important, even indispensable, to not have to log in to the system, although most thought it was not important. A few standard users thought it was very important to have a written user manual, but most did not. The security personnel seemed to disagree, where three persons thought this was important while four did not. This is a little surprising considering the security personnel's higher computer skills, but they also have much more complicated tasks to complete in the system, which might explain this result. Security personnel also seemed to be more interested in the system's appearance than the other two user groups. In general, security personnel found all usability properties significantly more important than the other two groups, but we are unable to identify the cause of this.

The users supplied their age, sex, department, computer skill level and type of user. While the type of user is the only categorization that is relevant to the KAS, it does not seem like this categorization is meaningful with respect to difference in opinions. The same is true for sex, age and department, only computer skill seems to notably affect the users opinions on what is important and how it is defined. We suspected that age should at least have an impact but it did not.

8.2.2 Analysis and Testing

Analysis of quality attributes

From the start the KAS has been designed with emphasis on functionality, usability and security. The ATAM shows that all these quality attributes negatively affects modifiability to some degree. It is therefore clear that some quality attributes oppose each other. While modifiability at the moment is considered important, it is also of lower priority than security and usability. While a higher level of modifiability is possible to achieve in time, it can only be modifiable to a certain degree before affecting security or usability. Thus future additions or changes in the system will require extensive technical documentation and skilled personnel. In other words, a compromise can be reached to some extent, but it will have to take the priorities into account. The design decisions made during the development process seems to have been the right ones, because the system conforms to the stated prioritization of qualities.

Testing quality attributes

Most of the parts of the system that were identified as lacking in quality, were linked to unhelpful error messages and unhandled exceptions. The testing phase identified the date and datetime fields with their awkward indata syntax requirements as the primary cause. These problems could be addressed with some fairly simple adjustments, for example dividing the text fields into logical parts or replacing them altogether with drop-down lists. This would positively affect several of the test results and eliminate a primary cause of user error, thus significantly improving the level of usability and security.

In general the user interface seems to be intuitive enough with the exception of the factors mentioned above. The level of security is generally high with a few easily remedied exceptions. The level of modifiability, however, is not too high and changes in the system can cause many ripple effects which are costly and time-consuming to repair.

8.2.3 Important quality attributes and definitions

The five most important properties to all users (except the three left in the "other" category) are listed here, with the most important on top:

- E11 - Protection of personal information
- E5 - The presentation of information on a well arranged way
- E10 - Protection against hackers
- E2 - Stability
- E7 - Error messages that are easy to understand

Surprisingly, protection against hackers is a primary concern by all users, we speculate that this is because people fear that the system stores personal information that can be stolen and/or misused. Security concerns, although underrepresented with properties in the questionnaire, are clearly highly important to all user groups.

The properties the users found least important are listed here, with the least important on top:

- E3 - Choice of colors
- E15 - User manual in paper form
- E4 - Appearance

Interestingly, the vast majority of the users considered user manuals (in paper form) to be unimportant. One might think that a user manual was preferred complementing a new computer system, but it seems that few would read it if it did. It is also interesting to note that the systems appearance, which we feel is quite important to it's intuitiveness and ease of use, was considered to be of such low importance.

The users mostly agreed on the definiton of quality attributes. Almost all users agreed that hiding prospective errors from the user is not usability, and most that showing all errors is. We find this very interesting, as Microsoft goes to great length to hide all errors from the user in Windows, presumably in the interest of increasing usability. Windows also has an abundance of help functions and wizards, and most users agree that this is not usability. The feature most people associated with usability was that a software system has well arranged menus.

The users have surprisingly similar definitions on security, performance and availability. Only minor differences can be observed between users with different computer skill level, and even less between user groups.

8.2.4 Generally

Through the comments people have submitted with their replies to the questionnaire (see appendix A) it is clear that our efforts in creating a new software system is appreciated, and for this we are grateful. Other comments and the data collected has been very helpful for both ourselves and the future administrators in identifying the pros and cons of the current system. This survey has produced much new information. In general we think the surveys and the test phase has been a success and we are satisfied with the results they provided, although some slightly more contradictory opinions amongst the users would have been appreciated.

Chapter 9

Conclusion and further work

9.1 Conclusion

We have tried to make a thorough survey of the preferences and opinions of all the future users of the KAS, and this seems to have been a success. Representatives for all user groups in the system have made their opinions clear. The opinions of our contacts at UNN are the basis for the requirements specification and their opinions are paramount. However all the user groups have been in agreement to a surprising extent as to what is important in the system. This is, of course, nothing but positive for the further development of the software, but it also limits the number of interesting conclusions that can be drawn about users and their supposedly categorical opinions. Although category that does seem to have an effect is the computer skill of the subjects, but only to a small degree. The subjects year of birth ranged from 1948 to 1981, but this did not seem to affect their answers. It can clearly be seen from the results which properties are important and which aren't. Also the fact that the part of the questionnaire which purpose was to capture the different users definitions of quality attributes seems to have been misunderstood, it is hard to draw any conclusions about definitions in general. Although the users were also here largely in agreement, and the charts provide some interesting results.

The analysis of the quality attributes based on the users priorities concluded that modifiability was a trade-off of usability and security, that is the latter negatively affected the former. This is however the necessary high-level design choice because of the importance of usability and security, and the level of modifiability will be dependent on this fact. The test phase uncovered some interesting sources of errors in the system, which can be fixed with little effort, but the quality was generally satisfactory. It also confirmed that modifiability could be better, and more work will be needed to achieve a highly modifiable system.

9.2 Further work

Some errors that undoubtedly would have been pointed to by the users have already been identified by the internal testing phase and can be fixed before a simulated user test is initiated. User testing is indeed the next step towards getting the KAS operational. We also feel we are on a more solid foundation with respect to the systems quality and the prioritization between differnt types of qualities. This, in turn, means that after a complete user test session, we have a clear picture of which further additions to the system we have to give priority. We also have a large pool of possible extensions and improve-

ments to choose from.

After the implementation is completed it is important to produce extensive technical documentation to enable other parties to make future changes or additions to the system without the help of the current developers.

Part V

Appendix

Appendix A

The questionnaire results

This appendix contains all the answers from the questionnaire.

We got 49 answers apart from some that were rubbish registrations. 36 of these 49 were from women and 13 from men. The year of birth of the participants varied in fact by a one year interval with almost every year from 1948 to 1981 represented. The participants classified themselves into user groups and the following is showing this classification:

User group	Count
Standard users	19
Department leaders	13
Security personnel	7
10 Other	10

A.1 Computer skills

This is how each participant qualified their own computer skills:

Skill	Count
Computers	1
Create documents in word processor	24
General understanding of MS Windows	19
General understanding of computers	4
Expert/professional	1

A.2 Important properties

How important are the following properties in a computer system like this?
The tables A.1 to A.20 shows the answer to this question. The "All users" for each property is shown in percentage, while the rest is the number(count) of people that classified the property at the respective importance.

	Unimportant	Less important	Very important	Indispensable
Standard user	1	4	12	2
Department leader	0	1	10	2
Security/watchmen	0	1	6	0
Other	1	1	6	2
All users	4	14	69	12

Table A.1: E1 - Fast response

	Unimportant	Less important	Very important	Indispensable
Standard user	0	0	8	11
Department leader	0	2	8	3
Security/watchmen	0	0	2	5
Other	0	0	5	5
All users	0	4	47	49

Table A.2: E2 - Stability

	Unimportant	Less important	Very important	Indispensable
Standard user	9	10	0	0
Department leader	5	6	2	0
Security/watchmen	0	7	0	0
Other	5	5	0	0
All users	38	57	4	0

Table A.3: E3 - Colors

	Unimportant	Less important	Very important	Indispensable
Standard user	9	9	1	0
Department leader	2	8	3	0
Security/watchmen	0	3	4	0
Other	2	7	1	0
All users	26	55	18	0

Table A.4: E4 - Appearance

	Unimportant	Less important	Very important	Indispensable
Standard user	0	1	15	3
Department leader	0	0	10	3
Security/watchmen	0	0	2	5
Other	0	0	8	2
All users	0	2	71	26

Table A.5: E5 - Well arranged presentation of information

	Unimportant	Less important	Very important	Indispensable
Standard user	0	4	14	1
Department leader	0	6	5	2
Security/watchmen	0	2	4	1
Other	0	4	6	0
All users	0	33	59	8

Table A.6: E6 - Able to use without previous knowledge

	Unimportant	Less important	Very important	Indispensable
Standard user	0	0	17	2
Department leader	0	1	9	3
Security/watchmen	0	0	6	1
Other	0	0	9	1
All users	0	2	83	14

Table A.7: E7 - Error messages that are easy to understand

	Unimportant	Less important	Very important	Indispensable
Standard user	0	2	15	2
Department leader	0	3	7	3
Security/watchmen	0	1	5	1
Other	0	0	8	2
All users	0	12	71	16

Table A.8: E8 - Self explaining tasks

	Unimportant	Less important	Very important	Indispensable
Standard user	0	3	15	1
Department leader	0	1	11	1
Security/watchmen	0	0	4	3
Other user	0	0	8	2
All users	0	8	77	14

Table A.9: E9 - Difficult to make mistakes

	Unimportant	Less important	Very important	Indispensable
Standard user	0	0	10	9
Department leader	0	2	6	5
Security/watchmen	0	0	2	5
Other	0	2	0	8
All users	0	8	36	57

Table A.10: E10 - Protection against hackers

	Unimportant	Less important	Very important	Indispensable
Standard user	0	0	8	11
Department leader	0	0	6	7
Security/watchmen	0	0	1	6
Other	0	0	1	9
All users	0	0	33	67

Table A.11: E11 - Protection of personal information

	Unimportant	Less important	Very important	Indispensable
Standard user	0	4	11	4
Department leader	0	2	7	4
Security/watchmen	0	1	4	2
Other	0	0	4	6
All users	0	14	53	33

Table A.12: E12 - Enough information to track possible abuse of the system

	Unimportant	Less important	Very important	Indispensable
Standard user	3	10	5	1
Department leader	2	6	4	0
Security/watchmen	1	4	2	0
Other	3	2	5	0
All users	18	45	33	2

Table A.13: E13 - Changing password often

	Unimportant	Less important	Very important	Indispensable
Standard user	5	7	7	0
Department leader	6	3	3	1
Security/watchmen	1	4	2	0
Other	4	3	2	0
All users	33	35	28	2

Table A.14: E14 - No login

	Unimportant	Less important	Very important	Indispensable
Standard user	10	6	2	1
Department leader	6	5	2	0
Security/watchmen	0	4	3	0
Other	8	1	1	0
All users	49	33	16	2

Table A.15: E15 - User manual in paper form

	Unimportant	Less important	Very important	Indispensable
Standard user	3	9	7	0
Department leader	0	7	5	0
Security/watchmen	1	1	5	0
Other	0	5	5	0
All users	8	45	45	0

Table A.16: E16 - Search after users and employees

	Unimportant	Less important	Very important	Indispensable
Standard user	4	13	2	0
Department leader	5	4	4	0
Security/watchmen	0	3	4	0
Other	3	5	2	0
All users	24	51	25	0

Table A.17: E17 - Show different kind of statistics

	Unimportant	Less important	Very important	Indispensable
Standard user	0	2	14	2
Department leader	0	4	8	1
Security/watchmen	0	1	4	2
Other	0	2	6	2
All users	4	18	62	14

Table A.18: E18 - To see the order status

	Unimportant	Less important	Very important	Indispensable
Standard user	0	4	11	4
Department leader	1	3	8	1
Security/watchmen	0	2	4	1
Other	1	1	8	0
All users	0	20	67	12

Table A.19: E19 - As many fields as possible should be pre-filled

	Unimportant	Less important	Very important	Indispensable
Standard user	0	3	12	4
Department leader	0	3	8	2
Security/watchmen	0	1	4	2
Other	0	2	6	2
All users	0	18	61	20

Table A.20: E20 - Have the option to regret choices

A.3 Quality definitions

What do you understand by the following words when it comes to any computer system (Usability, Performance, Security and Availability)?
The following tables are show in percentage. The participants could check out as many as they liked on these, and/or write their own definition. We received only a few answers that they wrote themselves. See these comments later in this appendix.

Usability	Standard	Leader	Security	Other
It looks like other systems	32	15	30	20
It has nice colors	0	0	14	0
It has well arranged menus	95	92	57	90
Many help and tutorial functions	11	15	14	0
Not many tutorial functions	68	15	14	60
That you need a user manual	32	15	14	20
That you don't need a user manual	47	46	29	70
That possible errors become hidden	0	8	0	0
That you easily could see all errors	53	69	29	90

Table A.21: What does usability mean to you?

Performance	Standard	Leader	Security	Other
The time it uses to finish work from pressing a button	47	69	43	70
How fast programs starts	42	54	43	50
How many programs that is open at the same time	26	23	14	20

Table A.22: What does performance mean to you?

Security	Standard	Leader	Security	Other
That a user is authenticated	58	69	71	70
That one is unable to access more info than necessary	79	92	71	90
That it is protected against hackers	58	85	71	70
That it is protected against viruses etc.	63	77	71	70
That personal information is kept confidential	63	62	57	70

Table A.23: What does security mean to you?

Availability	Standard	Leader	Security	Other
Information exist in several languages	0	0	14	0
Accessible from multiple locations, ie. office, home	26	23	14	0
That it detects errors, and is able to correct them	42	69	57	70
That it is not out of commission	74	69	57	90

Table A.24: What does availability mean to you?

A.4 Thoughts on system

If a computer system like this gets implemented, what do you think will happen?

a) Ordering key cards will be:
6 answered as easy/difficult as before
43 answered easier than before
0 answered more difficult than before

b) Writing an order will go:
4 answered as slow/fast as before
45 answered faster than before
0 answered slower than before

c) Orders with error will:
11 answered be same as before
38 answered be reduced
0 answered be increased

A.5 Comments

Quality attributes

Comments on usability:

- It is important that it is possible to make changes to an order, example if the time changes or the person is not supposed to have a card after all.

- Use templates

Comments on security:

- The safety have to be on top in a hospital.

- It is important that only someone have the opportunity to make an order. It would have been nice if you could see in the program who it is ordered to and who did that order.

A.5.1 General comments

We have removed all duplicate and equal comments

What is good with todays paper system?

- I have no problem with todays system
- It gives good exercise/condition
- Nothing compared to a user friendly computer system
- Easy to find in archives
- Nothing is good
- Who knows?
- Can't see any advantages. This would be possible to store electronically.
- Doesn't have safety on top
- Easy to handle
- Can't see any reason to keep it
- Well arranged, if everybody does what they are supposed to
- ????
- Compared to a computer system todays system sucks
- Can be done easier on a computer

What is bad with todays system?

- Lack of guidance, who have the right to order, changes on schema, etc.
- I have no problem with todays
- Cumbersome. You have to go and deliver it personally. No way to control that they received the orders
- Lots of paper laying around
- Takes to much time
- Difficult to control that everything is as it is supposed to. Is the order received?
- Not so well arranged, easy to make a mistake that could spoil the whole order.
- Delays, maybe the card isn't ready when the person needs it
- We have to make double copies before we send the order to prove that we actually sent it
- Not a trustable system
- Misunderstandings because of different ways to fill out the forms

- Orders sent beforehand "disappears"
- A lot of the id-cards have not been made even when copy of the order exists.
- Easy to forget if you actually delivered it, and it is a lot of walking to make sure you did
- Some orders "get lost"
- As everything else is done on a computer, it is apparent that this also should be done on a computer or else it would be very messy
- Can create a lot of errors

Other comments about a computer system like this?

- I am considered about safety, both on personal information and that intruders doesn't get access to cards
- Do it as fast as possible!!
- It is time to take us from the stone age to technology now!
- This is only a natural step towards the "paper-empty" hospital
- All orders that could be done on a computer is great!
- Very nice to get an electronic system
- Good luck to you guys!
- It would be nice not having to deliver it by yourself
- I hope that making orders would be easier
- I would like to have a list over persons having access to the different units/departments.
- I think it is a very good idea that this system is being considered as a replacement for the paper system
- It have to be easy, but secure

A.6 The questionnaire

A representation of the questionnaire in it's original norwegian form is presented here, with indexes for all questions. The original can be found here;
http://www.stud.ntnu.no/ halmo/diplom/surveyferdig.php

En enkel start

s1	Kjønn:	Kvinne Mann
s2	Fødselsår: eks. 1980	
s3	Hvilken avdeling jobber du på?	
s4	Hvilken type bruker er du?	Vanlig kortbestiller Avdelingsleder Vekter Annet
s5	Hvor mye kan du om data?	Data? Skrive dokumenter i Microsoft Word Generelle ferdigheter i windows Generell forståelse av pcer (Kan installere operativsystem) Ekspert (Kan bytte hardware, lage dataprogrammer)

Hvor viktige synes du de følgende egenskapene vil være i et slikt datasystem?

	Egenskaper	uviktig - litt viktig - veldig viktig - uunnværlig
e1	Rask respons	
e2	At systemet ikke er ustabilt	
e3	Fargevalg	
e4	Utseende	
e5	Informasjon presenteres på en oversiktlig måte	
e6	Må kunne bruke det uten noen forkunnskaper	
e7	Enkle, forståelige feilmeldinger	
e8	Arbeidsoppgavene er selvforklarende	
e9	Vanskelig å gjøre feil	
e10	Beskyttelse mot hackere	
e11	Godt vern av personlige opplysninger	
e12	Datasystemet burde ha nok opplysninger om brukeren slik at eventuelle misbrukere kan spores opp	
e13	Passord skiftes ofte	
e14	Man slipper å logge seg inn	
e15	Bruksanvisning i papirform	
e16	Mulighet for å søke etter brukere	
e17	Mulighet for å vise diverse statistikk	
e18	Man ser status på bestillingene	
e19	Ting som kan være forhåndsutfylt burde være det	
e20	Man har mulighet til å angre valg i menyer	
e21	Annet	

Velg fra listene og/eller forklar med egne ord. Hvis du kommer på noe vi ikke har nevnt her, syns vi det er ekstra interessant. Hva legger du i disse begrepene, når det er snakk om et hvilket som helst dataprogram?

b1	Brukervennlighet:	At det ligner på andre dataprogrammer?
b2		At det har fine farger?
b3		At det har oversiktlige menyer?
b4		At det har masse hjelpefunksjoner og veivisere?
b5		At det ikke har masse veivisere, men går rett på sak?
b6		At man har en brukerveiledning ved siden av?
b7		At man ikke trenger brukerveiledning?
b8		At eventuelle feil blir mest mulig skjulte?
b9		At man tydelig får se alle feil som oppstår?
b10		Annet
y1	Ytelse:	Hvor lang tid det går fra du trykker på noe til maskinen er ferdig å jobbe?
y2		Hvor raskt programmer åpnes?
y3		Hvor mange programmer du kan ha kjørende samtidig?
y4		Annet
s1	Sikkerhet:	At det kontrolleres at en bruker er den han/hun utgir seg for å være?
s2		At en bruker ikke har tilgang til mer informasjon enn nødvendig?
s3		At det er beskyttet mot hackere?
s4		At det er beskyttet mot virus o.l.?
s5		At opplysninger om deg selv forblir konfidensielle?
s6		Annet
t1	Tilgjengelighet:	At informasjonen finnes i flere språk?
t2		At det er mulig å komme inn på fra flere forskjellige steder, jobb, hjemme osv.?
t3		At det finner feil når de oppstår, og klarer å rette på disse?
t4		At det ikke er ute av drift?
t5		Annet

Tanker om dagens system

f1	Hvilke ulemper synes du det papirbaserte bestillingssystemet i dag har?	
f2	Hva er bra med det papirbaserte systemet?	
	Hvis man innfører et slikt datasystem, hva tror du vil skje?	
f3	Å bestille nøkkelkort blir	Lettere enn før
		Like lett/vanskelig som før
		Vanskeligere enn før
f4	Å skrive en bestilling vil gå	Raskere enn før
		Like raskt/tregt som før
		Tregere enn før
f5	Antall bestillinger med feil i vil	Minke
		Ikke endre seg
		Øke
f6	Har du noen andre tanker om å innføre et slikt datasystem i forhold til det nåværende papirbaserte systemet?	

Appendix B

Original system requirements

Here we present the finally requirements specification we followed during the development of the KAS. This specification is translated from its original form which was in Norwegian. To easily refer to these requirements in the report, we have numbered them R1, R2, R3 and so on.
Extra shifts/Part-time cards

- **R1** Employees supposed to order key cards must log into a system. It is essential that this system has high security, and that only those that are authorized as a person who can create orders is allowed into the system. Therefore it has to exist some kind of register with which employees is responsible for ordering.
- **R2** It is going to be one form for extra shifts (i.e. that is a form for ordering part-time key cards), and one system to handle all these orders. This ordering form must be self explaining so that one doesn't need to now detailed system specification to make use of it. It should mainly resemble the existing paper form, but some expansion will be needed.
- **R3** A list showing all orders from one department is required to avoid multiple orders of keys to the same person, or to check if an order is actually submitted. The orders in this list should be deleted automatically after a certain time, a month or so.
- **R4** The system must be able to deduce if the person who it is ordered to, already has a key card. In this way it knows if this person should have a card or if his or her existing card should be updated. Information on every employee must be stored in a database, and the identification of the actual person must be done with something other than social security number. One suggestion is to use a combination of date of birth, last name and first name as identificator. Either one have to complement this primary key with other fields to make it 100% unique, or accept the possibility of error in the system (which our contacts agreed on).
- **R5** Information about which department/unit the staff belongs to is required.
- **R6** A register of groups, that is different parts/jobs (i.e. a doctor), one can have in the different departments is required. When ordering key cards, one must choose both which department and which group the person belongs to.
- **R7** When the security personnel performs an order (registering a card on a person) it must be possible to "out-run" the system if the person already have a card that is

not registered as delivered. This means that if somebody forgets to press "innlevert" when the card is delivered, one must be able to update the system manually.

Id-cards to permanent employees

- **R8** Similar to the extra shift ordering, it has to be a registration form for permanent employees. Key cards to these people is id-cards with picture on it. Different reasons for registering an order like this are when a new employee arrives, one employee have changed positions, technical issues on the card or a person has lost his or her card. A database system is required for this orders too. Compared to the system for extra shift cards, this system could be quite similar with a few exceptions listed as their own requirements:

- **R9** Basically only one department is conducting orders to these employees, and that is the personnel department (personalavdelingen).

- **R10** Social security number can be used as identificator in the database if required.

- **R11** If the "Jobber til dato" field is filled out, the card is to be delivered at this date and therefore the card must be blocked for use after this date. If this field is not filled out, the card should be valid in infinite time until further messages.

- **R12** The system must log all of these orders, in the same way it does with the extra shift orders, when approved.

Employees from other places

- **R13** The system could provide a function from people outside(here it is meant outside the hospitals network) to access it. This is not that important and therefore this is not a priority. One possible solution to this could be to put the system on the Internet, but then it would affect the security.

Generally about these orders

- **R14** A field named "Bestillingsstatus" is required. This field will provide the users with status about their orders. Submitted orders starts with "venter" as condition in this field. This status is later changed to "godkjent/godkjent med tilbakemelding" or "avvist/avvist med tilbakemelding". This means that every order must contain a field "tilbakemelding" that gives the possibilities for the security personnel to write feedback to the person making the order in case something is missing/wrong. In this way the persons who order the cards is able to control the status and perform additional operations to get the orders approved. This will create a two-way communication channel between the orderer and the security personnel, and this could eliminate some possible errors.

- **R15** All logs must, in addition to card numbers, contain the key cards access level that defines if the cards is active and what they have access to. This must be updated every time an order is approved (i.e. the security personnel must be able to chose this), delivered or out of date

- **R16** The same person can have access to order cards to people on more than one department, if he or she works in several departments. There exists also several persons in each department that can order cards, and the system must consider this.

- **R17** It is not required to make a register that consists of all key cards that exists, both those that has a user and those lying on the shelf waiting to be used, even if this possibly makes sense from a database point-of-view. This is so because it would create a lot of extra work, and it would possibly become difficult to keep a large register updated. It is not necessary with more information about a key card so that one is able to find out who it belongs to.

  Logs

- **R18** All approved orders must be logged with card numbers. Today this is done in a paper book, which is difficult to keep track of when many orders exists. In addition to all information in todays log, it is necessary to store the orderers first name, last name, date of birth, department/group and time for ordering to register the person who sent the order.

- **R19** Logs that are older than one year should be deleted, except those with cards not delivered.

- **R20**It should be possible to:
  -See which cards that are not delivered at the time.
  -See which persons has ID-cards.
  -Get an overview of all logged happenings

  Administrator/Security

- **R21** The security personnel must be able to log in with username and password on their own computers.

- **R22** The administrators is the only user group that should be able to change and access all information in the system, and they should be the only ones able to give username/passwords/access levels in the system.

- **R23** The department leaders are an exception. They must be able to add orderers on their own department.

- **R24** The administrator must have his own pages were he could insert/edit/delete employees, orderers (the persons that can make an order), groups, firms, departments and access level.

- **R25** People that orders ID-cards, both those from "personalavdeling" and others, must be able to order to all departments, and see all orders done the last month.

- **R26** To keep the size on the database down, it is required that inactive employees (haven't had a card for a long time), is removed from the register. An option is to have a report that shows which employees have been inactive the longest.

  Security (This is extra, additional work)

- **R27** Retrieve the computer name(and/or windows username) to log which person was using the system at which time.

- **R28** The password function can have limitations, as example time limit, number of tries, consists of both numbers and characters

  Generally

- **R29** All schemes must be integrated into an electronically system that will ease up the work done with these orders. The system has big demands towards security, both when considering it as a system/database and with the authentication of persons. As development platform it is required that the system is programmed in a Microsoft Environment like ASP.NET and MS SQL.
- **R30** The system must keep its data separated from the security system at the hospital, and keep these data updated so that the administrator has access to this information an can update the existing security system manually.
- **R31** Our contacts at UNN wanted us to store as many things as possible in dropdown lists. In doing so we decrease the possibility of error inputs from the users.

Appendix C

Relational database schema

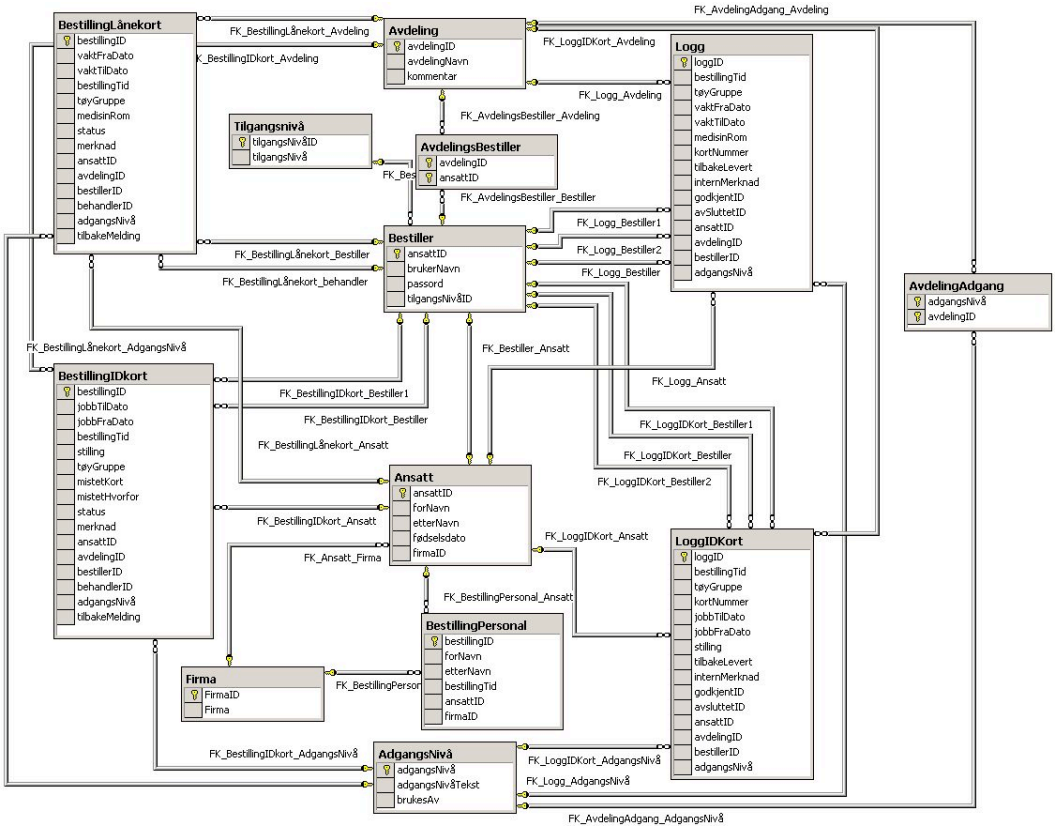

Figure C.1: Conceptual view of the database in MS SQL server

Appendix D

Glossary

Quality: The totality of characteristics of an entity that bear on it's ability to satisfy stated and implied needs [ISO 8402].

Attribute: A measurable physical or abstract property of an entity [ISO 14598-1].

Quality attribute: A mapping of a system's functionality into a software structure that is meaningful in measuring quality [2]. It is a categorization of software design desicions that affect a certain type of quality.

Design decision: A type software structure used in a computer system, for example client-server architecture or error detection algorithms [2]

External quality: The extent to which a product satisfies stated and implied needs when used under specified conditions [ISO 14598-1].

Internal quality: The totality of attributes of a product that determine its ability to satisfy stated and implied needs when used under specified conditions [ISO 14598-1].

User: An individual that uses the software product to perform a specific function [ISO 14598-1]. In this project it means anybody that will use the KAS to complete a task.

Measurement scale: A scale that constrains the type of data analysis that can be performed on it [ISO 14598-1].

Metric: A measurement scale and the method used for measurement [ISO 14598-1].

Bug: Coding error in software.

Debugging: The effort of removing the bugs.

Bibliography

[1] ISO/IEC JTC 1. Iso/iec 9126 information technology - software quality characteristics and metrics, 2001.

[2] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, second edition, 2004.

[3] Egon Berghout and Rini Van Solingen. *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*. McGraw-Hill Publishing, 1999.

[4] Marnie L. Hutcheson. *Software Testing Fundamentals*. Wiley Publishing, 2003.

[5] Glenford J. Myers, Tom Badgett, Todd M. Thomas, and Corey Sandler. *The Art of Software Testing*. John Wiley & Sons,Inc., second edition, 2004.

[6] Tor Stålhane. Etablering av måleplaner. *SPIQ notat*, June 1999.

[7] Clas Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering, An introduction*. Kluwer Academic Publishers, 2000.